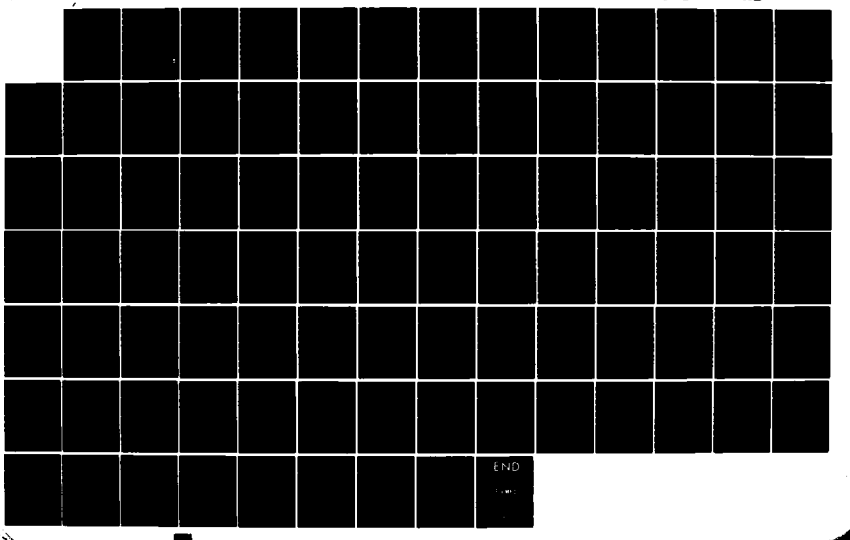
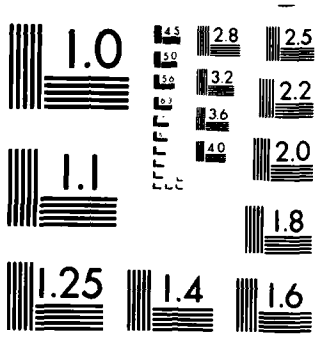
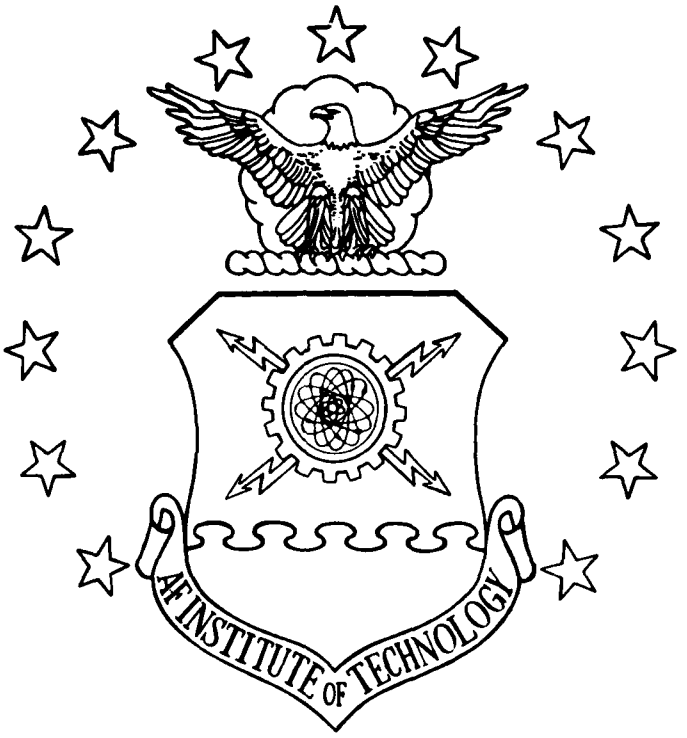
AD-A152 217 ADAPTIVE GRID GENERATION FOR NUMERICAL SOLUTION OF BURGER'S EQUATION(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING B D BOYD 1/1

UNCLASSIFIED DEC 84 AFIT/GAE/AA/84D-2 F/G 12/1 NL







ADAPTIVE GRID GENERATION FOR NUMERICAL SOLUTION OF BURGER'S EQUATION

THESIS

Bruce D. Boyd, B.S.
Second Lieutenant, USAF

AFIT/GAE/AA/84D-2

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GAE/AA/84D-2

ADAPTIVE GRID GENERATION FOR NUMERICAL SOLUTION OF BURGER'S EQUATION

THESIS

Bruce D. Boyd, B.S.
Second Lieutenant, USAF

AFIT/GAE/AA/84D-2

AFIT/GAE/AA/84D-2

ADAPTIVE GRID GENERATION FOR NUMERICAL SOLUTION OF BURGER'S EQUATION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Aeronautical Engineering

Bruce D. Boyd, B.S.

Second Lieutenant, USAF

December 1984

Acknowledgements

I would like to take this opportunity to thank those individuals who have helped me prepare this work. First, many thanks go to my thesis advisor, Dr. Sal Leone, for all his time and effort. Second, I would like to thank the other members of my committee, Dr. James K. Hodge and Dr. Dale Carlson, for their many helpful suggestions and advice. This work would not have been possible without the help of these experts in the field of Computational Fluid Dynamics.

Special thanks and appreciation go to my wife, Jo Ann, whose patience and understanding were invaluable.

Table of Contents

	Page
Acknowledgements	ii
List of Figures	v
List of Tables	vii
Abstract	viii
I. Introduction	1
Background	2
Literature Survey of Adaptive Grid Methods	4
Objectives	5
Overview	6
II. Mathematical Formulation	8
Transformation from Physical Plane to Computational Plane	8
Grid Generation	9
Adaptive Grid Criterion	10
Grid Control Function Evaluation	13
One-Dimensional Model Problem	15
Solution Procedure	16
Error Analysis	17
III. Discussion of Results	19
Test Case (Case 1)	19
Expected Results	24

Page

Grid Dependence on Re 24

Grid Dependence on $x_{\tau_{acc}}$ 25

Grid Dependence on u_ζ 26

Grid Dependence on N_p 26

Grid Dependence on Linear Region 26

Grid Dependence on Non-Negative P 27

Effect of Grid Refinement 27

Effect of Equations Used 27

Effect of Fixed Grid 28

Steady State Error 28

IV. Conclusions 30

V. Recommendations 31

Appendix A: Figures 33

Appendix B: Computer Program BURG Listing 66

Bibliography 76

Vita 78

List of Figures

Figure		Page
1.	Initial and Final Grid for Case 1	34
2.	Initial and Final Grid for Case 2	35
3.	Initial and Final Grid for Case 3	36
4.	Initial and Final Grid for Case 4	37
5.	Initial and Final Grid for Case 5	38
6.	Initial and Final Grid for Case 6	39
7.	Initial and Final Grid for Case 7	40
8.	Initial and Final Grid for Case 8	41
9.	Initial and Final Grid for Case 9	42
10.	Initial and Final Grid for Case 10	43
11.	Velocity Profile for Case 1 in the Physical Plane	44
12.	Velocity Profile for Case 1	45
13.	Velocity Profile for Case 2	46
14.	Velocity Profile for Case 3	47
15.	Velocity Profile for Case 4	48
16.	Velocity Profile for Case 5	49
17.	Velocity Profile for Case 6	50
18.	Velocity Profile for Case 7	51
19.	Velocity Profile for Case 8	52
20.	Velocity Profile for Case 9	53
21.	Velocity Profile for Case 10	54

Figure Page

22. Error Between Time Steps for Case 1 55

23. Error Between Time Steps for Case 2 56

24. Error Between Time Steps for Case 3 57

25. Error Between Time Steps for Case 4 58

26. Error Between Time Steps for Case 5 59

27. Error Between Time Steps for Case 6 60

28. Error Between Time Steps for Case 7 61

29. Error Between Time Steps for Case 8 62

30. Error Between Time Steps for Case 9 63

31. Error Between Time Steps for Case 10 64

32. Error Comparison, Constant and Adaptive Grids . 65

List of Tables

Table		Page
1.	Conditions for Cases 1 Through 11	20
2.	Results for Cases 1 Through 11	21

Abstract

An adaptive grid generation method is presented which is based on the reduction of truncation error in the computational plane. Grid adaption is achieved through minimization of the third derivative of the dependent variable in the computational plane. Burger's equation is solved because it represents typical nonlinear fluid-flow equations. An optimized Successive-Over-Relaxation method is used to solve Burger's equation using second-order-upwind differences for the convective term.

Results are presented for several cases which are compared to the results of a test case. The results show grid points are concentrated in high gradient regions.

ADAPTIVE GRID GENERATION FOR NUMERICAL SOLUTION OF BURGER'S EQUATION

I. Introduction

Finite-difference algorithms can be used to solve the governing equations of fluid mechanics and heat transfer when exact solutions are unavailable. In order to get a reasonable solution to a finite-difference algorithm, a suitable grid must be chosen. In most algorithms, a grid is selected initially and the grid point locations are held fixed during the calculation. The best grid point locations for a given problem are determined from a priori knowledge of the exact solution. Numerical grid generation schemes (see 1; 2) can be used to determine grid point locations without a priori knowledge of the exact solution. Adaptive grid generation schemes (see 3; 4), a class of numerical grid generation schemes, can also be used to determine grid point locations; however, the grid point locations are not held fixed during the calculation. Movable grid points can be concentrated in high gradient regions, a desirable goal.

The purpose of this thesis is to develop an adaptive grid method which reduces the truncation error in the computational plane. The method will be based on the

boundary layer-dependent grid generation method of Hodge and Stone (5). The grid spacing is determined such that the truncation error is minimized. The viscous Burger's equation will be used as a test of the adaptive grid method.

Background

Numerical grid generation schemes (including adaptive grid generation schemes) should satisfy the following: (1) Grid points should be surface oriented so boundary conditions can be applied at the surface without requiring interpolation. Thus, interpolation errors are not introduced into the solution. In an adaptive grid, surface-oriented grid points should move due to boundary motion. (2) High gradient regions, such as shocks and boundary layers, should be accurately resolved since large numerical errors can occur.

A surface-oriented grid can be generated by solving a system of elliptic partial differential equations, one equation for each coordinate direction. Thompson, Thames, and Mastin (6) develop a grid generation method based on the solution of Poisson equations. Grid point placement is controlled by the forcing functions (also called the grid control functions) in the Poisson equations. The grid point locations are found by interchanging the dependent and independent variables in the Poisson equations and solving the resulting equations.

In the past, gradients were resolved by placing many grid points in suspected high gradient regions. If the locations of high gradient regions were unknown, a fine grid was used over the entire domain. Since more points are present in the fine grid, more computer time is needed to find the solution. An adaptive grid can be generated such that grid points will move into high gradient regions as the gradients in the computed solution develop. Two benefits accrue from the use of an adaptive grid. First, fewer grid points are needed since the grid points will move toward high gradient regions, thus reducing computer time. Second, truncation error in the computed solution is reduced. Large truncation errors can occur in high gradient regions. If grid points are concentrated in high gradient regions, the grid spacing is reduced, thereby reducing truncation error. If a suitable grid adaption criterion is selected, such as minimizing the magnitude of the derivatives in the truncation error, the truncation error is reduced.

Hodge and Stone (5) develop a boundary layer dependent grid based on the Blasius solution which minimizes truncation error. Truncation error is reduced by placing grid points such that increments in the velocity are constant. Therefore, the second and higher-order derivatives of the velocity in the computational plane which contribute to truncation error are minimized. Grid

point placement is accomplished by modifying the grid control functions in the grid generation equations of Thompson et al. Since the Blasius solution was used to place the grid points, this type of grid generation is valid only where boundary layer theory is valid. The method of Hodge and Stone is said to be a first-order method. The grid generation method developed in this thesis is similar except for the following: (1) The present method is said to be second order because the third derivative of the velocity in the computational plane is minimized. (2) The grid is not dependent on boundary layer theory.

Examples of adaptive grid methods follow.

Literature Survey of Adaptive Grid Methods

Gradients in the dependent variables can be used to move grid points as the computed solution develops. Dwyer, Kee, and Sanders (7) place grid points in proportion to gradients that appear in the developing computed solution. Grid skewness is one weakness of this method.

Error minimization is another adaptive grid technique. Pierson and Kutler (8) minimize a measure of local truncation error with respect to the transformation parameters. Anderson and Rai (9) induce a velocity in each grid point with respect to every other grid point based on the gradient at each grid point compared to the average gradient. Grid points are attracted to regions where local

gradients are higher than the average gradient and repelled from regions where local gradients are lower than the average. Error is reduced since grid points are concentrated in high gradient regions. Brown (10) minimizes truncation error in the computational plane by generating a grid such that the third derivative of the dependent variable in the computational plane is minimized.

Variational techniques are also used in adaptive grid generation. Saltzman and Brackbill (11) utilize a variational technique to minimize a linear combination of grid smoothness, orthogonality, and area variation.

Other criteria can be used to adapt a grid to the developing solution. The adaptive grid criterion used by Ghia, Ghia, and Shim (12) is the minimization of the coefficient of the convective term in the transformed flow equations. This technique is developed for high-Reynolds number flows.

Objectives

The objective of this thesis is to develop an adaptive grid method which reduces truncation error in the computational plane. The grid control function used in the grid generation equation is systematically determined by minimizing the third derivative of the dependent variable in the computational plane. The method is based on the boundary layer-dependent method of Hodge et al. and is similar to the method of Brown. The present method differs

from the method of Brown in the method of solving the grid generation equation, and the determination of the grid control function.

A one-dimensional model problem (the viscous Burger's equation) is used to test the method. Boundary conditions are imposed such that a boundary layer-type velocity profile is produced. Grid points are expected to be concentrated in the boundary layer.

Overview

Section II contains the following: the transformation used to transform derivatives in the physical plane to a computational plane; the one-dimensional grid generation equation; the criterion used to adapt the grid; the determination of the grid control function; the one-dimensional model problem; the solution procedure; and the error analysis used to evaluate the method.

The criterion used to adapt the grid is the minimization of the third derivative of the velocity in the computational plane. Minimization is achieved by fitting the computed solution with a second-order polynomial. The third derivative is related to the grid control function, thus the grid points are moved as the third derivative is minimized. The grid control function is determined from a Newton-Raphson iterative scheme which minimizes the third derivative.

Section III contains the discussion of results. Several cases were run to discover the effect of various input parameters on the method. In most cases, grid points were concentrated in the boundary layer; however, truncation error was not minimized in all cases.

The conclusions are found in Section IV. The method is very sensitive to initial conditions; therefore, the method may only be useful for a few combinations of input parameters.

Recommendations for further work include using a different criterion for grid convergence; updating the grid periodically instead of every time step; and using a different method for determining the grid control function. All recommendations are found in Section V.

II. Mathematical Formulation

Transformation from Physical Plane to Computational Plane

Just as cylindrical coordinates are used instead of cartesian coordinates to simplify the solution of certain problems, such as flow around a cylinder, transforming variables in the physical plane to variables in a computational plane may simplify the solution of a numerical algorithm. Arbitrarily-shaped regions in the physical plane can be uniquely transformed to a rectangular region in the computational plane with uniform grid spacing, assuming the Jacobian of the transformation is non-zero. The shape and spacing of the grid in the computational plane will remain constant even if the grid points are moving in the physical plane.

For a one-dimensional time-dependent problem, the computational variables, ζ and τ, are functionally related to the physical variables, x and t, by

$$\tau = t \tag{1a}$$

$$\zeta = \zeta(x,t) \tag{1b}$$

Derivatives in the physical plane can be transformed in the computational plane using the chain rule yielding

$$u_x = u_\zeta / x_\zeta \tag{2a}$$

$$u_{xx} = (u_{\zeta\zeta} - \frac{x_{\zeta\zeta}}{x_\zeta} u_\zeta) / x_\zeta^2 \tag{2b}$$

$$u_t = u_\tau - (\frac{x_\tau}{x_\zeta}) u_\zeta \tag{2c}$$

where the subscripts denote partial differentiation. The dependent flow variable, u, can represent velocity, temperature, pressure, etc. The complexity of Eqs. (2) do not outweigh the advantage of using a rectangular grid with uniform spacing.

Grid Generation

In this study, a one-dimensional grid is used for ease of computation and computer economy.

The one-dimensional grid generation equation used in this thesis is developed in Brown (10) and has the form

$$x_{\zeta\zeta} + P x_\zeta = 0 \tag{3}$$

where P, the grid control function, is a function of ζ and τ. Brown used an implicit optimized Successive-Over-Relaxation (SOR) method to solve Eq. (3). A second-order-upwind difference was used for x_ζ based on the sign of P. A central difference was used for $x_{\zeta\zeta}$. The solution of Eq. (3) using the given finite differences produces

truncation error of order $(P\Delta\zeta)^2$ (13), where $\Delta\zeta$ is the grid spacing in the computational plane.

A more accurate method of solving Eq. (3) is given by Lick and Gaskins (13). A Unified Difference Relation (UDR) is developed which is an exact representation of Eq. (3) for a constant P. The UDR corresponding to Eq. (3) is given by

$$x_{i+1} - (1+e^{-P})x_i + e^{-P}x_{i-1} = 0 \qquad (4a)$$

If P takes on large negative values, an overflow condition may result when Eq. (4a) is solved on a computer. Thus, for P negative, the UDR corresponding to Eq. (4a) is given by

$$e^{P}x_{i+1} - (e^{P}+1)x_i + x_{i-1} = 0 \qquad (4b)$$

Equations (4) are solved using a tridiagonal solver.

Adaptive Grid Criterion

Any finite-difference representation of a derivative has truncation error associated with it due to the truncation of the Taylor series used to define the derivative. For example, a central-difference representation of a first derivative in the computational plane is

$$u_\zeta = (u_{i+1}-u_{i-1})/2\Delta\zeta + \text{T.E.} \qquad (5)$$

where the truncation error, T.E., is given by

$$\text{T.E.} = -\frac{(\Delta\zeta)^2}{3!} u_{\zeta\zeta\zeta} - \frac{(\Delta\zeta)^4}{5!} u_{\zeta\zeta\zeta\zeta\zeta} + \ldots \qquad (6)$$

Note that Eq. (5) is said to be second-order accurate and note that $\Delta\zeta$ is an arbitrary constant and for convenience will be taken as unity for the remainder of this work.

In order to reduce the truncation error in the computational plane, the method developed in this thesis reduces the magnitude of $u_{\zeta\zeta\zeta}$.

If u can be exactly represented by a second-degree polynomial in the computational plane, $u_{\zeta\zeta\zeta}$ and all higher-order derivatives are identically zero. Globally approximating u by a second-degree polynomial may not be feasible. A local second-degree-polynomial approximation to u will minimize $u_{\zeta\zeta\zeta}$ locally, thus u can be approximated by

$$u(\zeta) = a_{2_i}\zeta^2 + a_{1_i}\zeta + a_{0_i} \qquad (7)$$

To discover what values of u contribute to the local truncation error, approximate $u_{\zeta\zeta\zeta_i}$ by a central difference, thus

$$u_{\zeta\zeta\zeta_i} \sim u_{\zeta\zeta_{i+1}} - u_{\zeta\zeta_{i-1}} \qquad (8a)$$

Using central differences to approximate the second derivatives in Eq. (8a) yields

$$u_{\zeta\zeta\zeta_i} \sim u_{i+2} - 2u_{i+1} + 2u_{i-1} - u_{i-2} \qquad (8b)$$

Thus, local truncation error is, in a general way, dependent on u at surrounding grid points. This suggests that the computational plane can be divided up into several subregions, each of which satisfies Eq. (7). To determine the coefficients a_{2_i}, a_{1_i}, and a_{0_i}, a least-squares curve fit is performed at every point using data from the surrounding grid points at the previous iteration level. The least-squares curve fit used in this thesis is taken from an algorithm in Conte and de Boor (14). The optimum curve fit is found by minimizing a sum of polynomials of degree two and below.

Each subregion of the computational plane consists of a number of data points, N_p, which must contain at least three points and not more than the total number of grid points. If N_p is equal to the total number of grid points, only one set of coefficients a_{2_i}, a_{1_i}, and a_{0_i} is found for the entire computational plane. If N_p is less than the total number of grid points, a balanced curve fit is performed at each point; i.e., if $N_p = 5$, the values of u used to calculate the curve fit will be taken from the $i \pm 2$, $i \pm 1$, and i locations. At or near the boundaries, the curve fit will be unbalanced.

The grid control function must now be determined such that the grid points will be placed in such a way as to force u to locally fit a second-order polynomial. Thus

$u_{\zeta\zeta\zeta}$ will be minimized locally and truncation error will be reduced.

Grid Control Function Evaluation

Since grid point locations are determined from the value of P (see Eqs. (3) and (4)), a relation is developed involving $u_{\zeta\zeta\zeta_i}$, P, and a_{2_i} which will be minimized in a Newton-Raphson scheme. Starting with an equation of the form of Eq. (8a), $u_{\zeta\zeta\zeta_i}$ can be approximated by a difference of second derivatives. Thus

$$u_{\zeta\zeta\zeta_i} \sim u_{\zeta\zeta_{i_1}} - u_{\zeta\zeta_{i_2}} \tag{9}$$

where $u_{\zeta\zeta_{i_1}}$ is derived from Eq. (2a) and $u_{\zeta\zeta_{i_2}}$ is derived from Eq. (7). From Eq. (2a)

$$u_\zeta = u_x x_\zeta \tag{10}$$

Differentiating Eq. (10) with respect to ζ yields

$$u_{\zeta\zeta} = u_{xx} x_\zeta^2 + u_x x_{\zeta\zeta} \tag{11}$$

Equation (11) is used to define $u_{\zeta\zeta_{i_1}}$. To determine $u_{\zeta\zeta_{i_2}}$, differentiate Eq. (7) with respect to ζ twice, thus

$$u_{\zeta\zeta_{i_2}} = 2a_{2_i} \tag{12}$$

Substituting Eqs. (11) and (12) into Eq. (9) yields the equation to be minimized:

$$F = u_{xx}x_{\zeta}^{2} + u_{x}x_{\zeta\zeta} - 2a_{2} = 0 \tag{13}$$

To get F as a function of P, divide Eq. (13) by x_{ζ} and note $x_{\zeta\zeta}/x_{\zeta} = -P$ from Eq. (3), thus F becomes

$$G(P) = u_{xx}x_{\zeta} - Pu_{x} - \frac{2a_{2}}{x_{\zeta}} = 0 \tag{14}$$

Values of P can be found by using a Newton-Raphson scheme which has the form

$$P^{(S+1)} = P^{(S)} - G^{(S)}/(G')^{(S)} \tag{15}$$

where S denotes the iteration level and the prime denotes differentiation with respect to P. Substituting Eq. (14) into Eq. (15) gives

$$P^{(S+1)} = P^{(S)} - (u_{xx}x_{\zeta} - Pu_{x} - \frac{2a_{2}}{x_{\zeta}})/(-u_{x}) \tag{16}$$

Utilizing Eqs. (2) for u_{x} and u_{xx} and after some simplification, Eq. (16) becomes

$$P^{(S+1)} = P^{(S)} + (u_{\zeta\zeta} - 2a_{2})/u_{\zeta} \tag{17}$$

Eq. (17) is used to determine P at each grid point. Note if P is converged, the second-degree-polynomial approximation of u is satisfied. Thus, the optimum grid is produced and truncation error is minimized.

One-Dimensional Model Problem

To test the adaptive grid method given above, the one-dimensional unsteady viscous Burger's equation is used. The viscous Burger's equation was chosen because it simulates nonlinear fluid flow equations. Burger's equation also has an exact solution thus the truncation error can be determined directly.

In nonconservative form, the viscous Burger's equation in the physical plane is

$$u_t + uu_x = u_{xx}/Re \tag{18a}$$

where Re is the Reynolds number. The boundary conditions used in this thesis are

$$\begin{aligned} u(-1,t) &= 1 \\ u(0,t) &= 0 \end{aligned} \tag{18b}$$

and the initial conditions are

$$\begin{aligned} u(0,0) &= 0 \\ u(x,0) &= 1, \quad x < 0 \end{aligned} \tag{18c}$$

With these conditions, a velocity profile similar to a boundary layer profile develops near the x=0 boundary. The thickness of this boundary layer-type profile is proportional to 1/Re. The exact steady state solution is given by

$$u(x) = - \tanh(Re\ \frac{x}{2}) \tag{19}$$

The solid line in Fig. 12 shows u(x) for Re = 1000. Transforming Eq. (18a) to the computational plane yields

$$u_\tau + \left(\frac{u-x_\tau}{x_\zeta}\right) u_\zeta = \frac{1}{Re\ x_\zeta} \left(\frac{u_\zeta}{x_\zeta}\right)_\zeta \tag{20}$$

This viscous term can be broken into two parts, one of which can be included in the convective term, thus

$$u_\tau + \left(\frac{u-x_\tau}{x_\zeta} + \frac{x_{\zeta\zeta}}{Re\ x_\zeta^3}\right) u_\zeta = \frac{u_{\zeta\zeta}}{Re\ x_\zeta^2} \tag{21}$$

Eq. (21) is used by Brown (10). Both (20) and (21) are used in this thesis.

An implicit optimized SOR method is utilized to solve Eqs. (20) and (21). First-order-backward differences are used for u_τ and the grid speed, x_τ; a second-order-upwind difference is used for u_ζ based on the sign of its coefficient; central differences are used for the viscous terms and x_ζ.

Solution Procedure

A listing of the computer program BURG used to solve the viscous Burger's equation using an adaptive grid is found in Appendix B. The algorithm used in the program consists of the following steps:

Step 1. Input an initial guess for the grid control function, P.

Step 2. Solve the appropriate grid generation equation, either Eq. (3) or Eq. (4) for the grid point locations.

Step 3. Solve the appropriate equation, either Eq. (20) or Eq. (21), for the velocity, u, at each grid point.

Step 4. Perform a least-squares curve fit of the computed solution at every grid point to determine a_{2_i}.

Step 5. Using Eq. (17), generate the values of the grid control function, P, at every grid point.

Step 6. Repeat steps 2 through 5 until steady state is reached. Steady state is assumed to be reached when the average difference in the computed solution between two time levels is less than a specified tolerance; i.e.,

$$\frac{1}{I} \sum_{i=1}^{I} |u_i^n - u_i^{n-1}| < \varepsilon \qquad (22)$$

where I is the total number of grid points, n denotes the time level, and the tolerance, ε, has a value of 10^{-6} in this thesis.

Error Analysis

Three types of error can be used to evaluate the success of this method: truncation error, fit error, and steady state error. Since the exact solution is known, truncation error can be calculated as the difference between

the computed solution and the exact solution. Fit error is defined as the difference between the computed solution and the least-squares curve fit. Average values of truncation error, $T.E._{ave}$, and fit error, $F.E._{ave}$, are calculated as

$$T.E._{ave} = \frac{1}{I} \sum_{i=1}^{I} |u_i - u_{exact}| \tag{23a}$$

$$F.E._{ave} = \frac{1}{I} \sum_{i=1}^{I} |u_i - u_{fit}| \tag{23b}$$

Steady state error is defined as the difference in the computed solution between the final two time levels; i.e.,

$$S.S.E. = |u_i^{n_T} - u_i^{n_T-1}| \tag{23c}$$

where n_T denotes the time level at convergence. Steady state error gives an indication of local convergence.

The optimum grid is defined when the average grid speed is less than a specified tolerance, $x_{\tau_{acc}}$, thus

$$\frac{1}{I} \sum_{i=1}^{I} |x_{\tau_i}| < x_{\tau_{acc}} \tag{23d}$$

When condition (23d) is met, the grid is held fixed and the computed solution is allowed to iterate until condition (22) is met. When the computed solution has converged, the fit error is assumed minimized, thus the truncation error is assumed minimized.

III. Discussion of Results

To test the adaptive grid method developed in this thesis, several cases were run on a CDC Cyber 845 computer to determine the effects of various input parameters. The conditions used for the different cases are given in Table 1. Results are given in Table 2 and also in Figures 1 through 32. (All figures are found in Appendix A.) Figures 1 through 10 show the initial and final grids. An idea of how well the grid adapted can be seen in these figures. Figures 11 through 21 show the exact and computed velocity profiles. How well the method matched the exact solution can be seen in these figures. Figures 22 through 32 show the steady state error at each grid point in the computational plane.

Test Case (Case 1)

The results of a test case (Case 1) are used for comparisons with the results of the other cases. The parameters are Reynolds number, Re; average grid speed tolerance, $x_{\tau_{acc}}$; minimum u derivative allowed, u_ζ; the number of points used in the least-squares curve fit, N_p; the number of grid points in which u is approximated by a linear function. Also, the values of the grid control function P; the total number of grid points, I; and the

TABLE 1

CONDITIONS FOR CASES 1 THROUGH 11

Case	Re	$x_{\tau_{acc}}$	u_{ζ}	N_p	Linear Region	P≥0?	I	$P_{initial}$	Grid Equation(s)	Solution Equation
1	1,000	.005	.1	5	5 pts.	No	21	.1	(4)	(20)
2	1,000	.005	.1	5	10	No	21	.1	(4)	(20)
3	1,000	.005	.1	21	5	No	21	.1	(4)	(20)
4	1,000	.0001	.1	5	5	No	21	.1	(4)	(20)
5	1,000	.005	.1	5	5	No	51	.1	(4)	(20)
6	1,000	.005	.01	5	5	No	21	.1	(4)	(20)
7	1,000	.005	.1	5	5	Yes	21	.1	(4)	(20)
8	100	.005	.1	5	5	No	21	.01	(4)	(20)
9	10,000	.005	.1	5	5	No	21	.1	(4)	(20)
10	1,000	.005	.1	5	5	No	21	.1	(3)	(21)
11	1,000	--	.1	--	5	--	21	.1	(4)	(20)

Notes:

1. Case 1 is the test case.
2. Case 11 does not use an adaptive grid.

TABLE 2

RESULTS FOR CASES 1 THROUGH 11

Case	$T.E._{ave}$	$F.E._{ave}$	Total No. of Time Steps, n_T	No. of Points in the Boundary Layer	Total CPU Time (seconds)
1	.01168	.05590	8	6	1.082
2	.002985	.01935	26	11	4.930
3	.04158	.1336	22	7	4.074
4	.001389	.003228	45	8	8.328
5	.001024	.0005713	7	9	3.166
6	.004002	.02276	18	6	2.400
7	.004687	.01895	10	6	1.500
8	.004127	.06411	11	7	1.665
9	.009522	.05132	7	5	1.158
10	.001911	.02685	7	2	.823
11	.002021	--	3	2	.449

equations used to generate the grid and the form of the viscous Burger's equation are investigated.

Reynolds number, Re, is varied to determine how well the grid adapts as the boundary layer thickness changes and the test-case value of Re is 1000.

The value of the grid convergence criterion, $x_{\tau_{acc}}$, is varied to determine whether the optimum grid is found. Fit error is expected to be minimized for a small value of $x_{\tau_{acc}}$ and the test-case value is .005.

The value of u_ζ must be limited to non-zero values since u_ζ is in the denominator of the second term on the right-hand side of Eq. (17). Test-case value of u_ζ is chosen as .1 because u_ζ = .1 represents ten points in the boundary layer if u is assumed linear there.

The number of points used in the least-squares curve fit, N_p, is varied to determine the dependence of truncation error on the computed solution at the surrounding grid points. The test-case value of N_p was chosen as 5 based on Eq. (8b).

Early results showed that very few grid points were concentrated in the boundary layer. Positive values of the grid control function, P, are needed to move points into the boundary layer. In order to force P to be positive in the boundary layer, u is approximated as a linear function in a region next to the x=0 boundary. In this region, a_{2_i} is zero, thus P will become positive if $u_{\zeta\zeta}$ is negative,

since u_ζ is negative in the boundary layer (see Eq. (17)). Note that $u_{\zeta\zeta\zeta}$ is still minimized if u is approximated by a linear function. The number of points in the linear region is five in the test case.

If P changes sign somewhere in the domain, a relatively large spacing is produced between two grid points. This spacing is a result of a negative P causing grid points to move away from the boundary layer and a positive P causes grid points to move toward the boundary layer. This spacing is undesirable because a smooth transition is needed between regions of high gradient and regions of low gradient for an accurate solution (10). To eliminate the large spacing, P can be constrained to non-negative values. In the test case, P is allowed to take on negative values.

The total number of grid points, I, is varied to determine if grid refinement leads to a more accurate solution. A larger number of grid points in the boundary layer is expected as the grid is refined and the test-case value of I is 21.

The form of the equations used or the method used to solve them may have an effect on the solution. Equations (4) are used for grid generation in the test case, and Eq. (20) is used to solve the viscous Burger's equation. As a comparison, an optimized SOR method is used to solve both Eq. (3), for the grid point locations, and Eq. (21), for the solution of Burger's equation.

An initial guess for P at every point is needed to start the program. An initial P of zero produces a cartesian grid. However, early results showed that the grid did adapt very well using an initial P of zero. Thus, a positive value of P was used initially to produce a grid which tends toward the boundary layer.

An error comparison is made between the test case and a fixed-grid case. A fixed grid is achieved by holding P constant at each grid point. The fixed grid corresponds to the initial grid of the test case.

Expected Results

The computed solution is expected to match the exact solution, thus truncation error is reduced, and grid points are expected to concentrate in the boundary layer. As a rule of thumb, five to ten grid points are necessary to accurately resolve the gradient.

Results and discussion of the cases tested follow.

Grid Dependence on Re

Case 8 was used to determine whether more grid points can be forced into the boundary layer if a small Re is used. The Reynolds number for Case 8 was 100 and seven points were placed in the boundary layer as compared to six for the test case (Fig. 19). Average truncation error was less than half the average truncation error of the test case (see Table 2), due to a smaller gradient. The

Reynolds number for Case 9 was 10,000 and this Reynolds number was used to see the effect of using a larger gradient on the grid adaption. Five points were placed in the boundary layer (Fig. 20) and truncation error was not significantly reduced. As Reynolds number is increased, more grid points are expected to be concentrated in the boundary layer; however, for the grid adaption used, fewer grid points were concentrated in the boundary layer as Re is increased.

Grid Dependence on $x_{\tau_{acc}}$

The value of $x_{\tau_{acc}}$ in Case 4 was .0001 and more time steps were required for a converged solution than for the test case. Average fit error was less than one-tenth the fit error for Case 1. This fact suggests a more optimum grid was produced in Case 4. However, the grid produced contained a very large grid spacing between two points (Fig. 4). The velocity profile (Fig. 15) shows that the curve fit is not very good in the vicinity of the large grid spacing. Thus, the third derivative is not minimized near the large spacing and truncation error is not minimized there.

The assumption that a converged grid leads to minimum truncation error is not correct.

Grid Dependence on u_ζ

In an attempt to allow P to be determined by actual slopes rather than limited slopes, the value of u_ζ in Case 6 was reduced to .01. Very few values of u_ζ are above .1 so reducing the limit on u_ζ allows more values of the slope to affect the solution. The final grid produced for Case 6 (Fig. 6) had an extremely large grid spacing between two points which caused the computed solution to converge very slowly. Note that the curve fit is not very good near the large spacing, thus truncation error is not minimized there (Fig. 17).

Grid Dependence on N_p

Poor results were achieved when N_p was allowed to include all grid points (Case 3). One of the largest values of average truncation error was produced by Case 3. The results show a parabola is not a good fit to a hyperbolic tangent.

Grid Dependence on Linear Region

The linear region had an effect on the shape of the final grid in all cases where a linear approximation for u is made (Figs. 1, 2, 4, 6, 7, 8, 10). In all these cases, the final grids exhibit a relatively large grid spacing between two grid points. These spacings occur at or near the edge of the linear region. This suggests that a_2 may be causing P to change signs since a_2 is zero in

the linear region and non-zero outside the region. Because of the large spacing, a good curve fit cannot be performed, thus truncation error is not minimized (Figs. 12, 13, 15, 17, 18, 20, 21).

In Case 2, u was approximated as a linear function in a 10-point region near the x=0 boundary. The number of points in the boundary layer for Case 2 is 11. Increasing the size of the linear region is a good method of increasing the number of points in the boundary layer; however, a good curve fit is not produced at the edge of the linear region (Fig. 2).

Grid Dependence on Non-Negative P

As noted above, a large spacing in the grid is produced near the edge of the linear region where P changes sign. To eliminate this large spacing, P was limited to non-negative values in Case 7. A better grid is produced (Fig. 7) but the curve fit is not very good at the edge of the linear region (Fig. 18) thus truncation error is not minimized.

Effect of Grid Refinement

When the total number of grid points, I, is allowed to increase, better resolution of the gradient is expected; i.e., a proportionate number of grid points is expected to move into the boundary layer. In Case 5, I is 51 and the results show lower truncation error and fit error, but only

nine grid points were placed in the boundary layer. Twelve to 15 grid points were expected to move into the boundary layer since more than twice the number of grid points were available for adaption. Not much adaption took place in Case 5 (Fig. 5), thus similar results could be expected from a fixed grid with the same number of points. Note, however, the increase in CPU time over Case 1.

Effect of Equations Used

In Case 10, Eq. (3) was used to generate the grid and Eq. (21) was used to solve the viscous Burger's equation. Both were solved implicitly using an optimized SOR method and poor results were found. Only two points were placed in the boundary layer. Although average truncation error is low, truncation error is not minimized for the points in the boundary layer (Fig. 21).

Effect of Fixed Grid

A constant value of P was used in Case 11 to hold grid points fixed. The results were poor when compared to adaptive grids. The average truncation error was low because only two points were in the boundary layer where the largest truncation errors occur.

Steady State Error

Steady state error is defined by Eq. (24d). Figures 22 through 31 show steady error in the computational plane. The regions of highest error correspond to the linear

regions in most cases. This suggests that the computed solution is not fully converged in the linear region. A nearly uniform distribution of error was expected.

Figure 32 shows a comparison of steady state error between Cases 11 and 1. Note that for the constant grid case (Case 11), only two points were in the boundary layer while six points were in the boundary layer in the adaptive case (Case 1). Thus, even though the fixed grid case has less error, the boundary layer is not resolved.

IV. Conclusions

The following conclusions can be drawn from the results presented:

1. The adaptive grid method concentrates grid points in the boundary layer. For Reynolds number 1000, an average of about seven grid points is placed in the boundary layer.

2. Truncation error is not minimized everywhere in the computational plane because an optimum grid is not produced.

3. The method is very sensitive to initial conditions. A change in one input parameter produces drastic changes in grid shape and velocity profile.

4. Final grid spacing is affected by the linear region where u is approximated by a linear function. A relatively large spacing between two grid points is produced at the edge of the linear region because the grid control function, P, changes sign there.

V. Recommendations

Several changes to the method are recommended in order to produce an optimum grid which minimizes truncation error.

The grid is assumed converged when the average grid speed is less than $x_{\tau_{acc}}$. A better convergence criterion is the fit error. Determination of the grid control function, P, is based on the least-squares curve fit, thus if P is converged, then u satisfies the second-degree-polynomial approximation and fit error is minimized. A truly optimum grid is produced when P is converged.

In the present method, the grid is updated every time step. Better adaption may be achieved by periodically updating the grid. Updating in the grid is similar to finding the computed solution on a fixed grid, then using that solution as an initial guess for the next solution.

The linear region affects the final grid spacing. However, approximating u as a linear function near the x=0 boundary is necessary to attract grid points to the boundary layer. When sufficient grid points are in the boundary layer, a linear approximation is no longer needed, thus the grid can be allowed to smoothe itself and allow better curve fits to be performed.

Average error is used to determine if the computed solution is converged (see condition (22)). Using maximum error as a convergence criterion may give better results and a more uniform distribution of steady state error.

Lastly, another method to determine the grid control function, P, may be necessary. The Newton-Raphson used in this thesis can only minimize the third derivative of u in the computational plane. A method which eliminates the third derivative may reduce truncation error significantly.

Appendix A: Figures

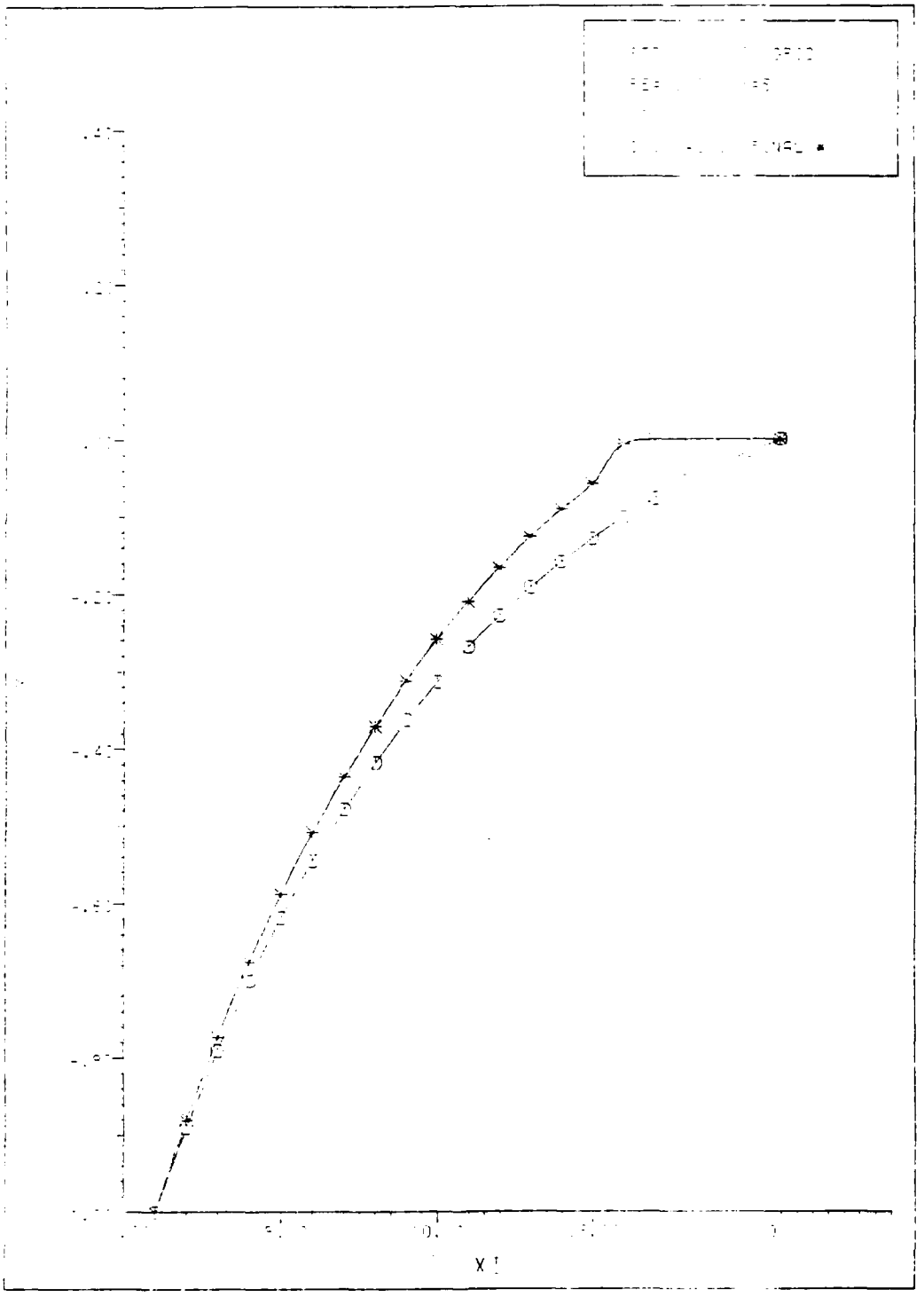
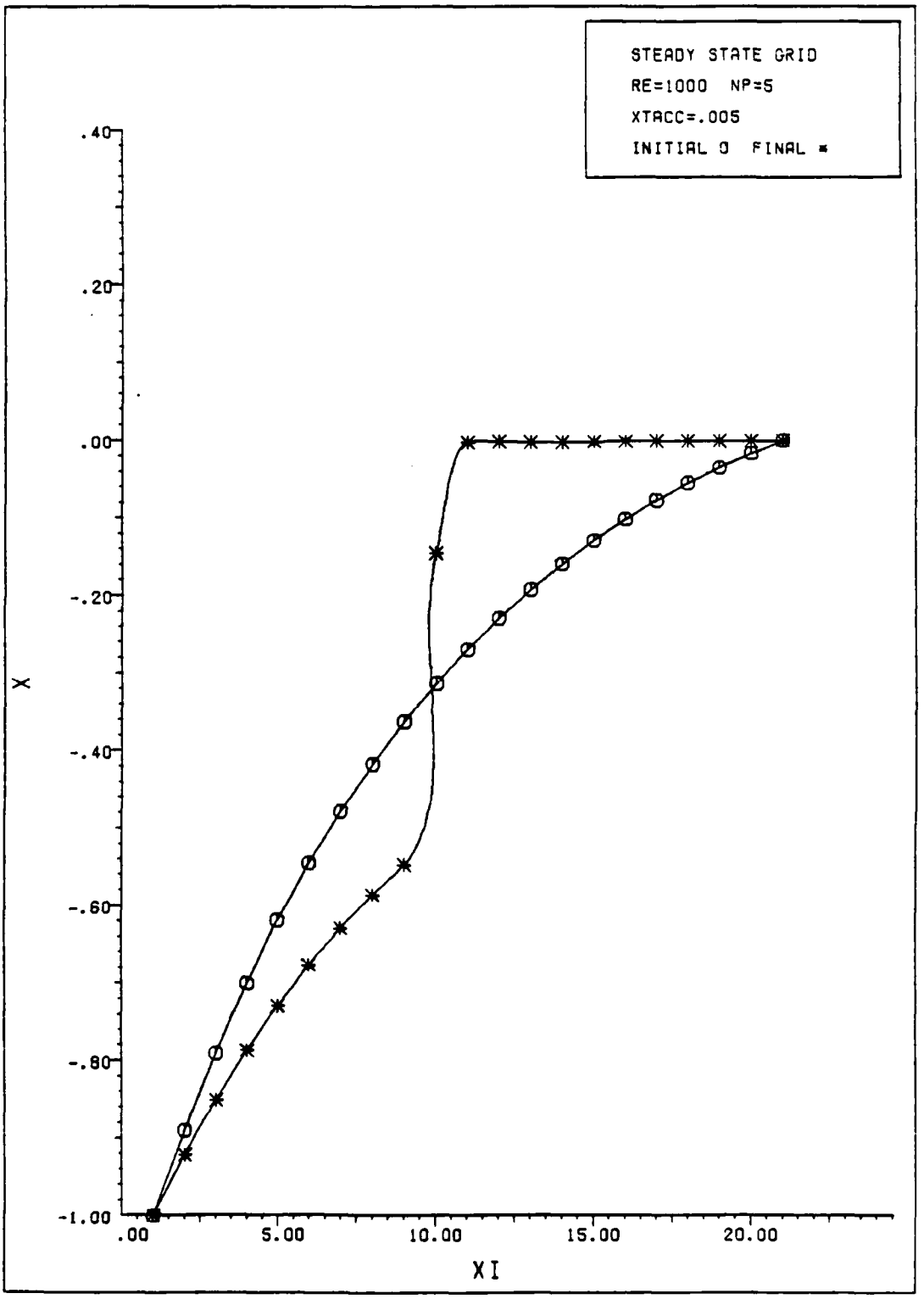
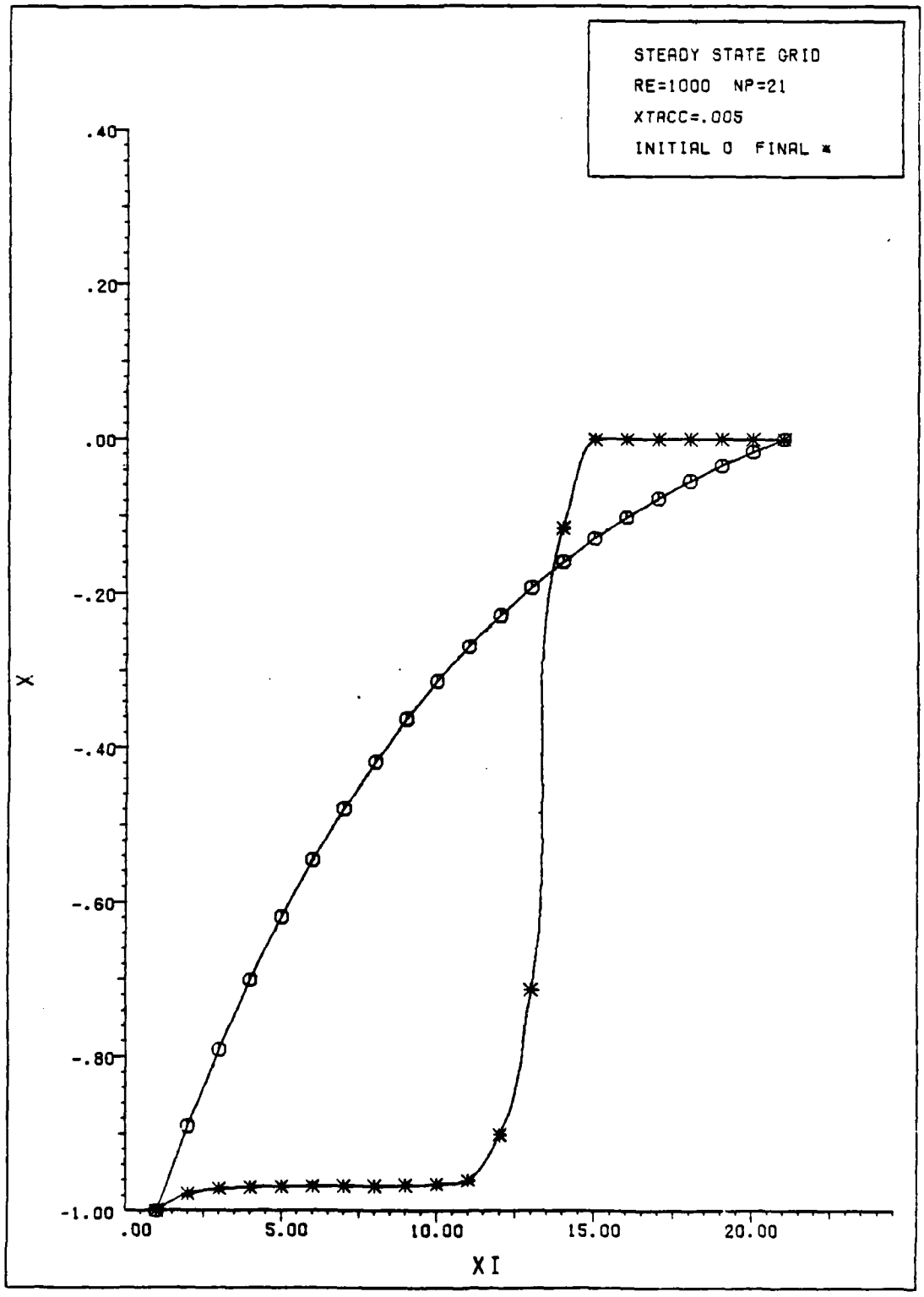


Figure 1



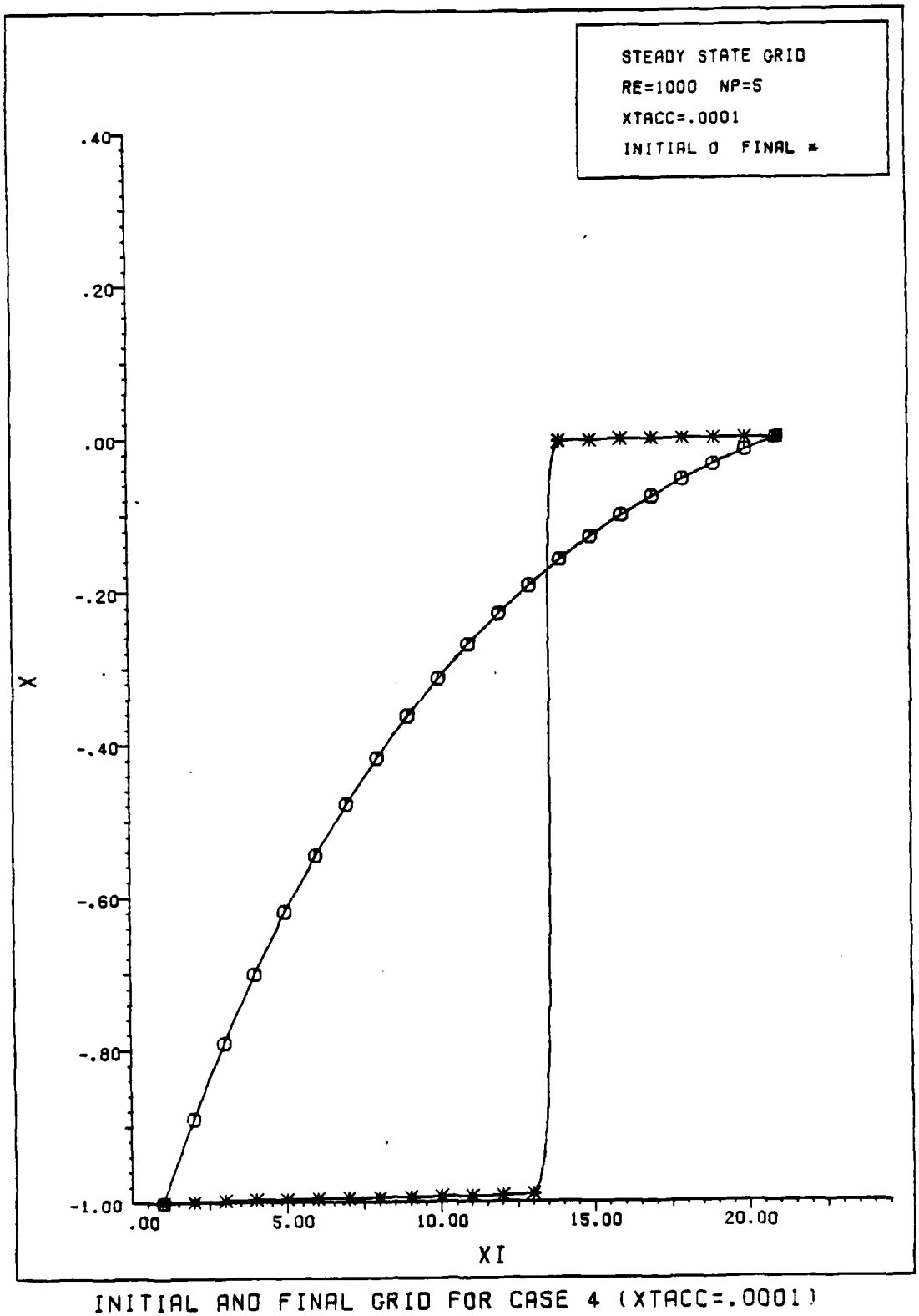
INITIAL AND FINAL GRID FOR CASE 2

Figure 2



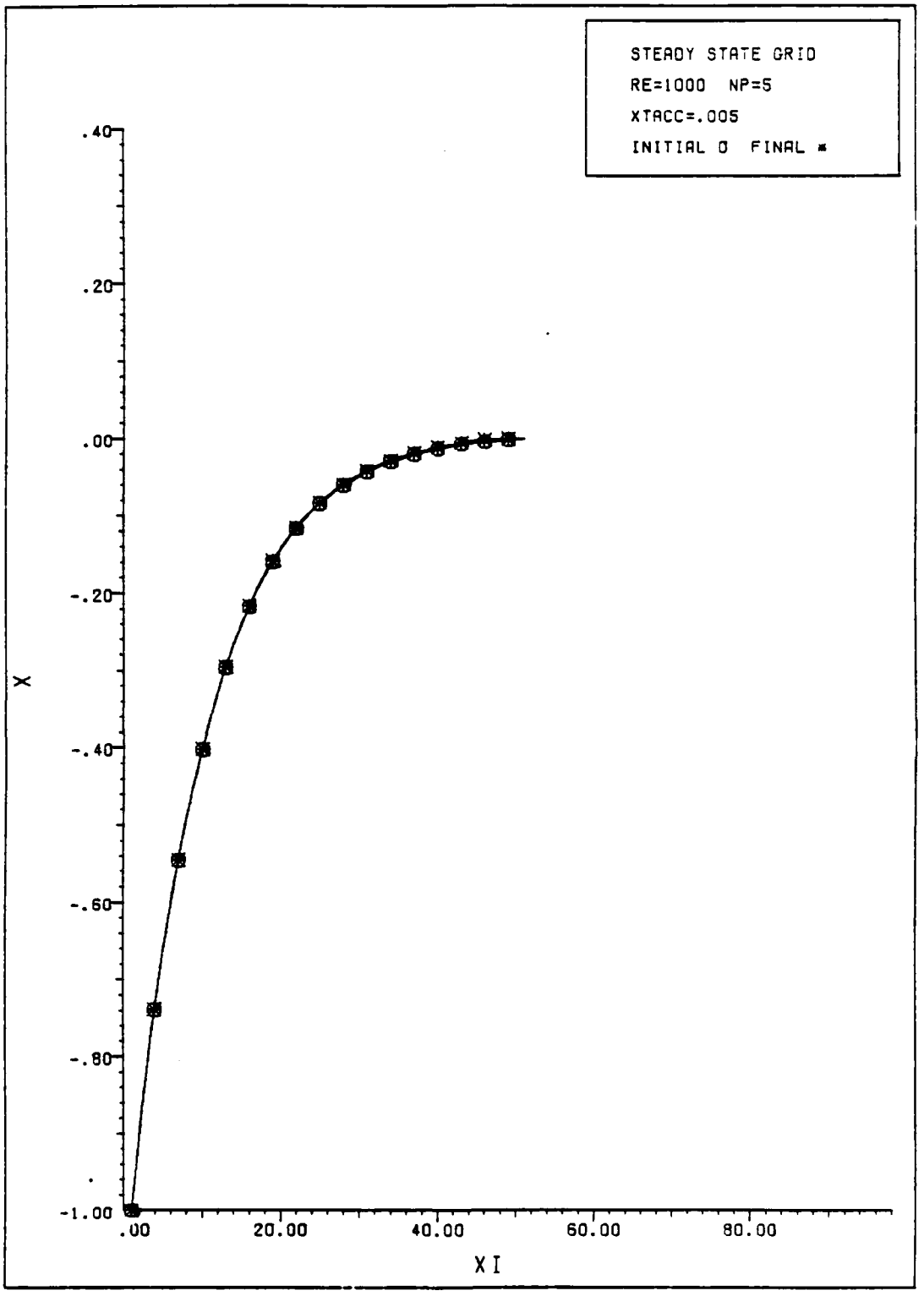
INITIAL AND FINAL GRID FOR CASE 3 (NP=21)

Figure 3



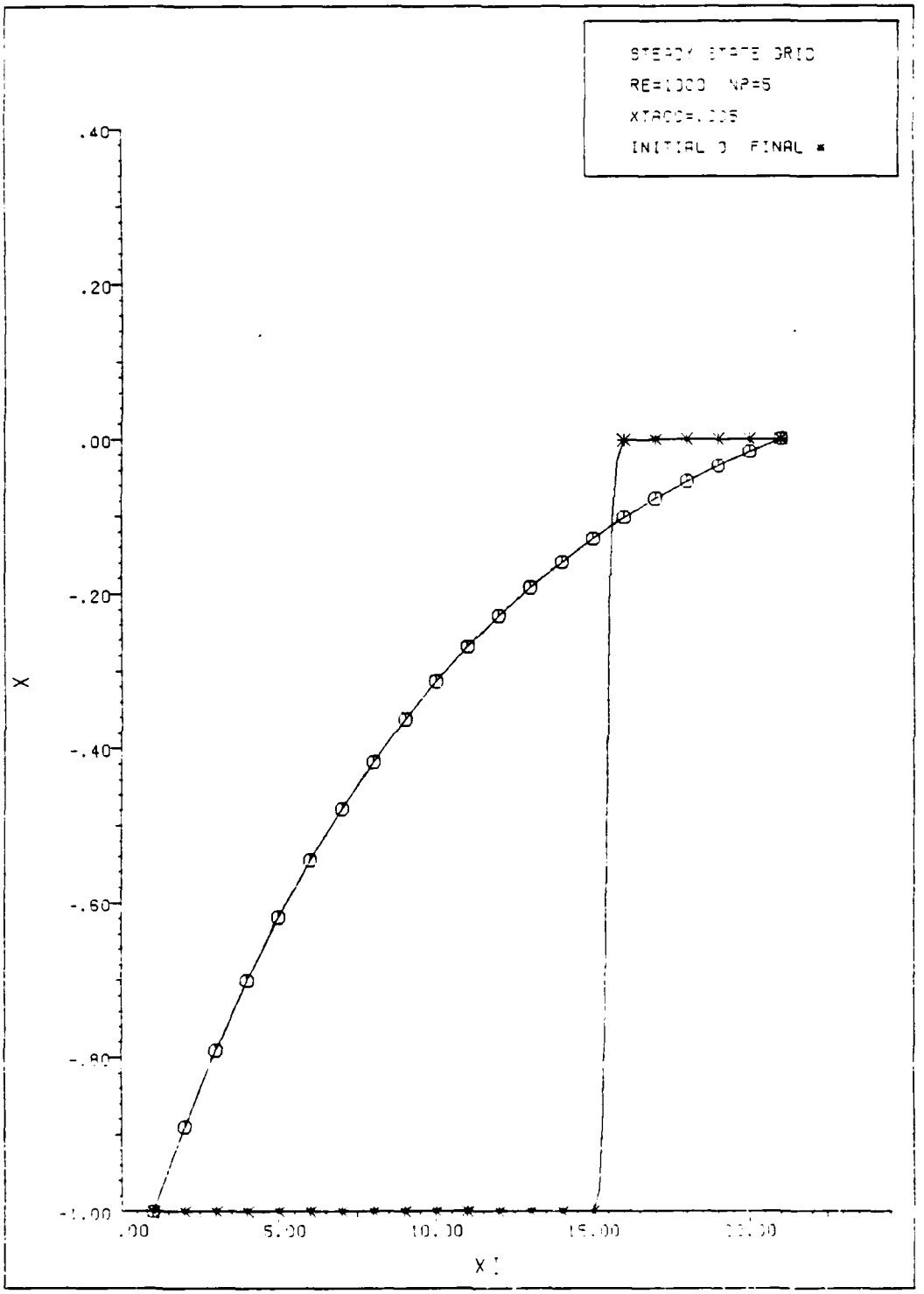
INITIAL AND FINAL GRID FOR CASE 4 (XTACC=.0001)

Figure 4



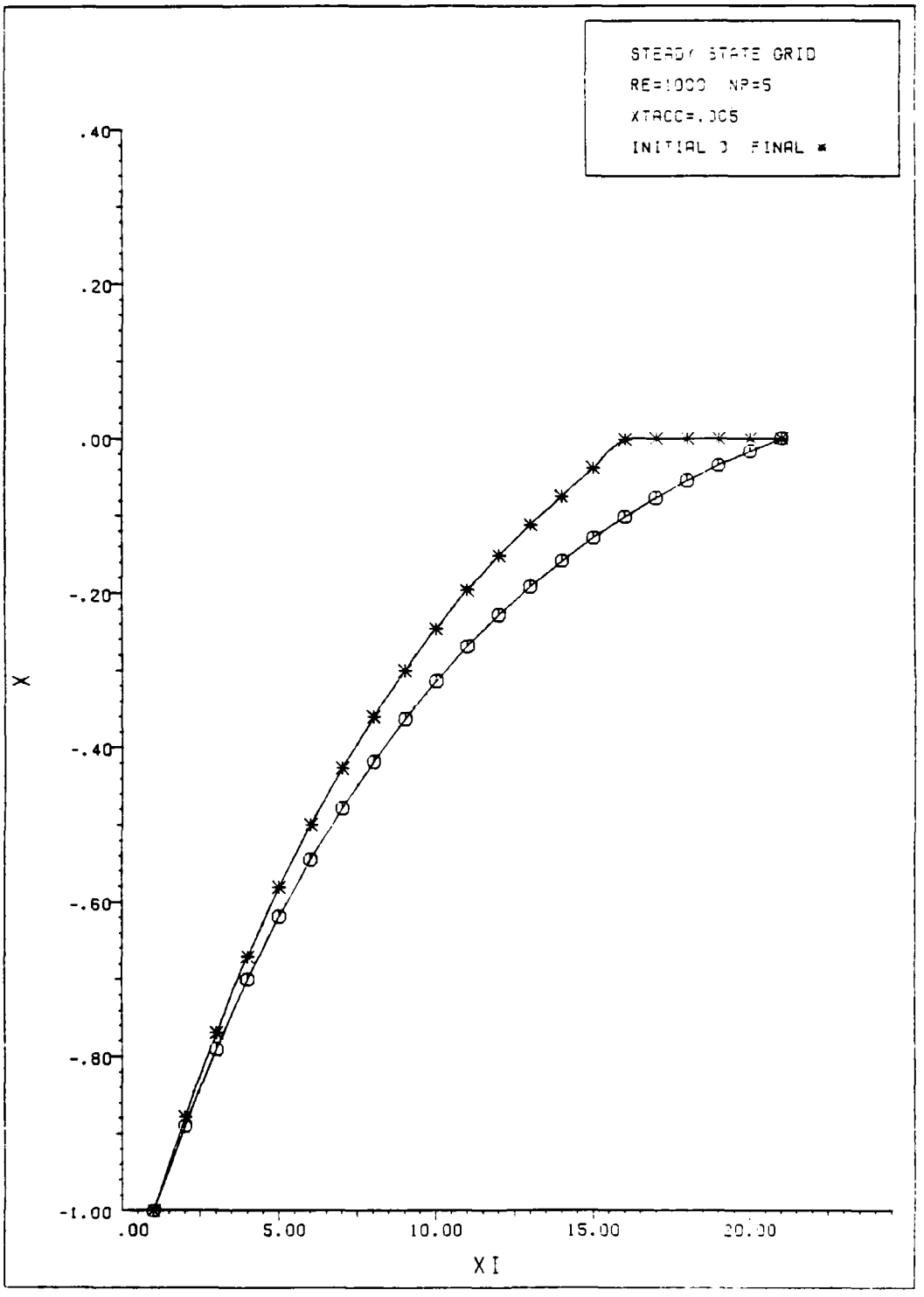
INITIAL AND FINAL GRID FOR CASE 5 (51 GRID POINTS)

Figure 5



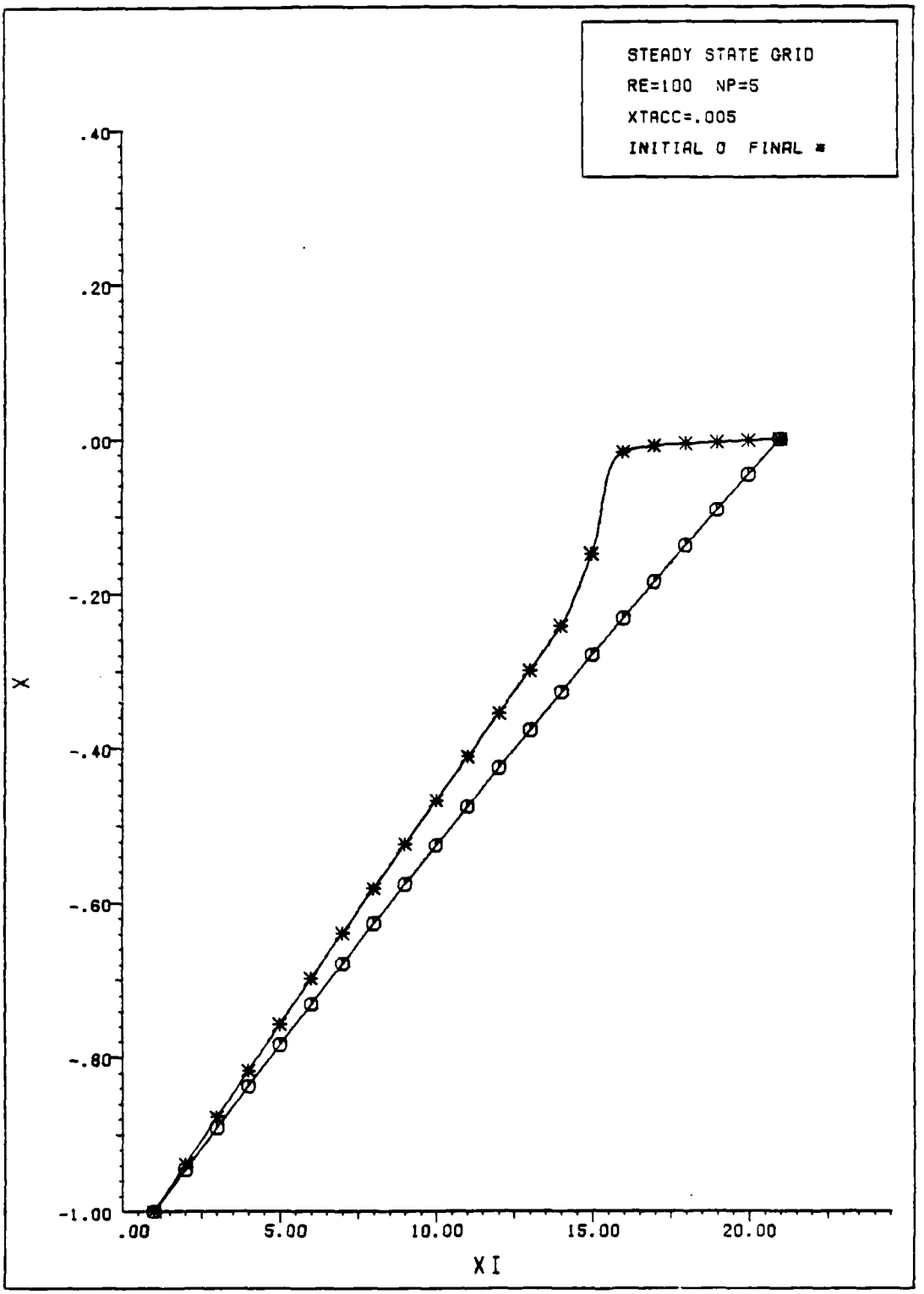
INITIAL AND FINAL GRID FOR CASE 5 - SLOPE = .01

Figure 6



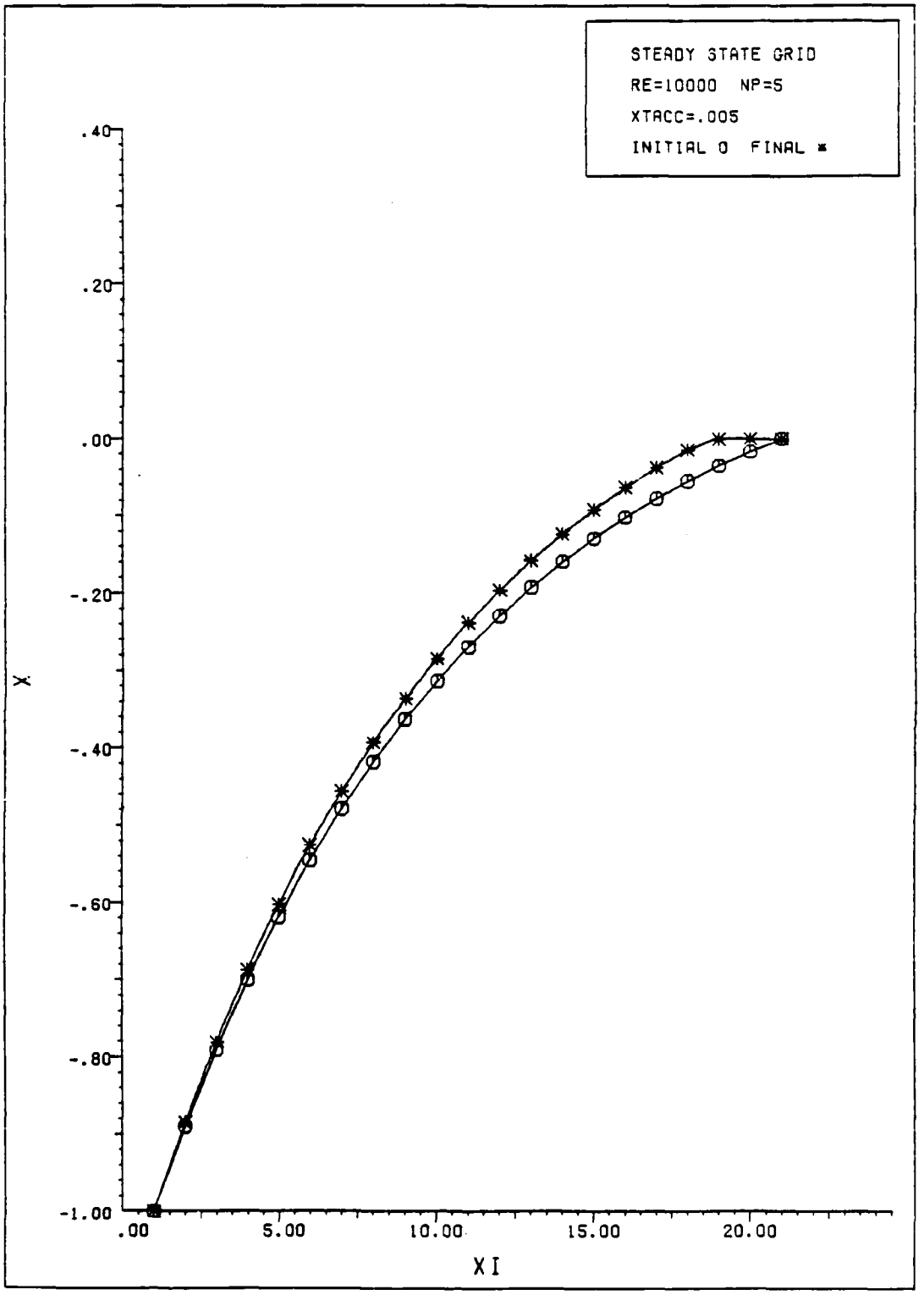
INITIAL AND FINAL GRID FOR CASE 7 (P ≥ 0)

Figure 7



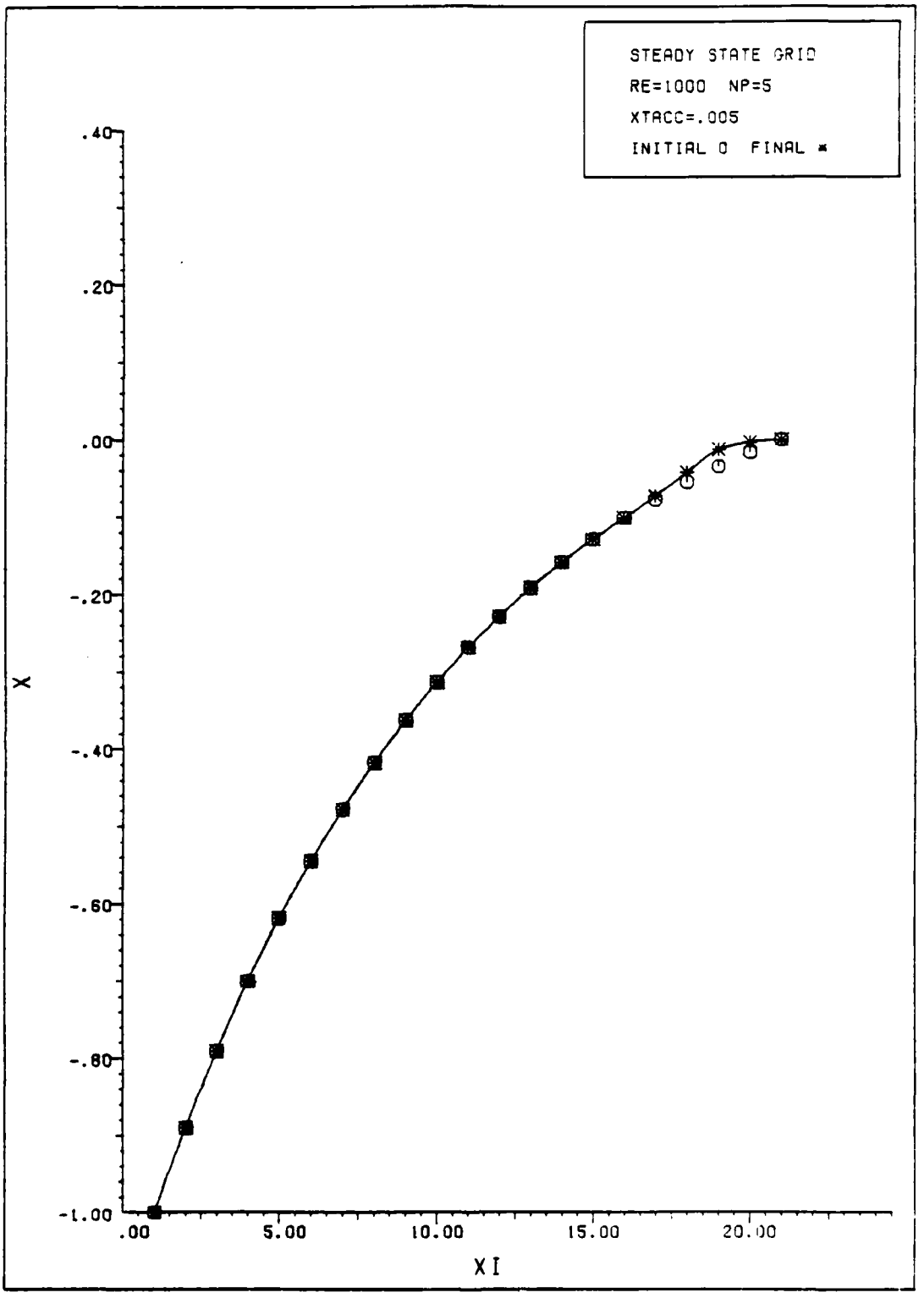
INITIAL AND FINAL GRID FOR CASE 8 (RE=100)

Figure 8



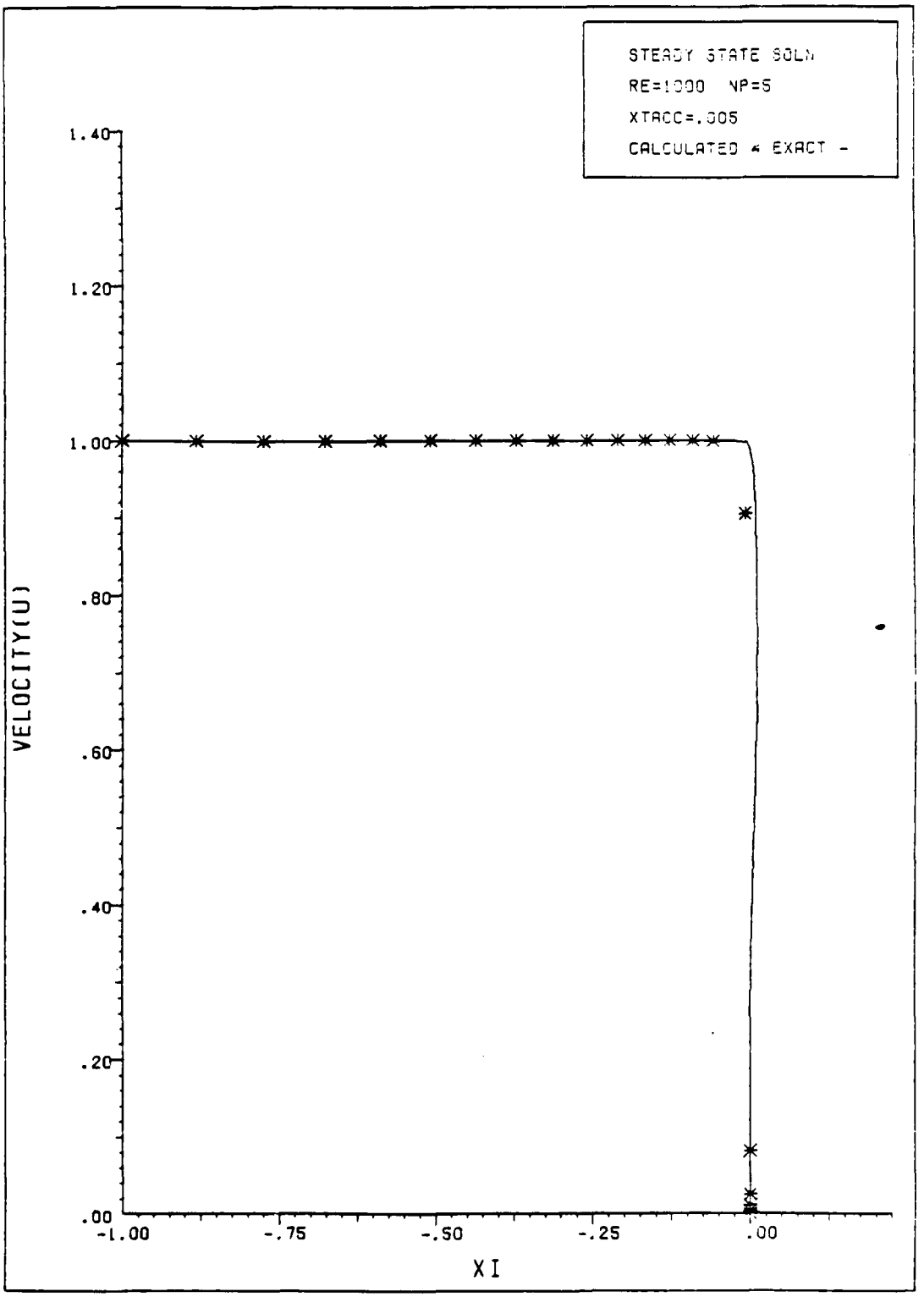
INITIAL AND FINAL GRID FOR CASE 9 (RE=10000)

Figure 9



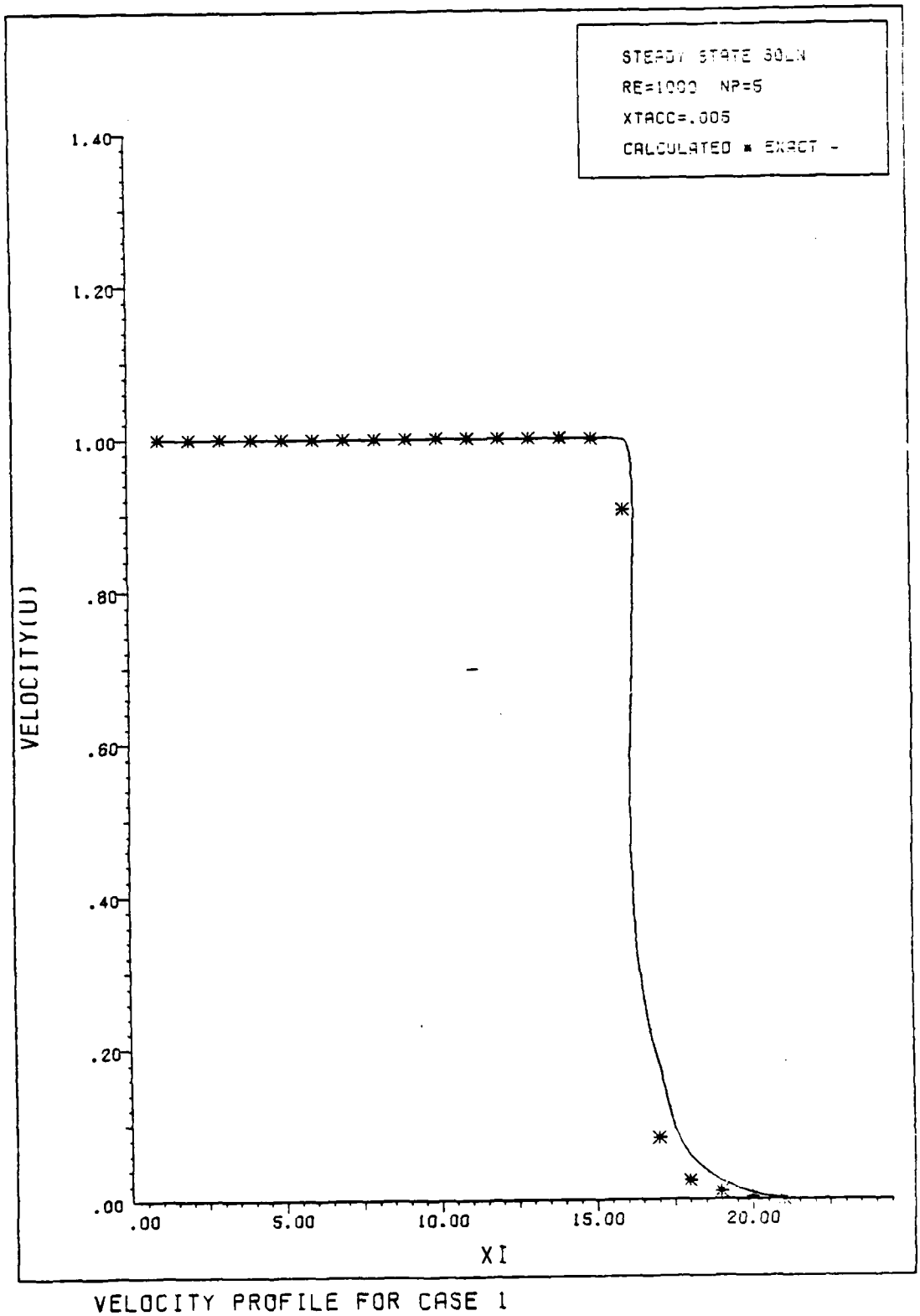
INITIAL AND FINAL GRID FOR CASE 10

Figure 10



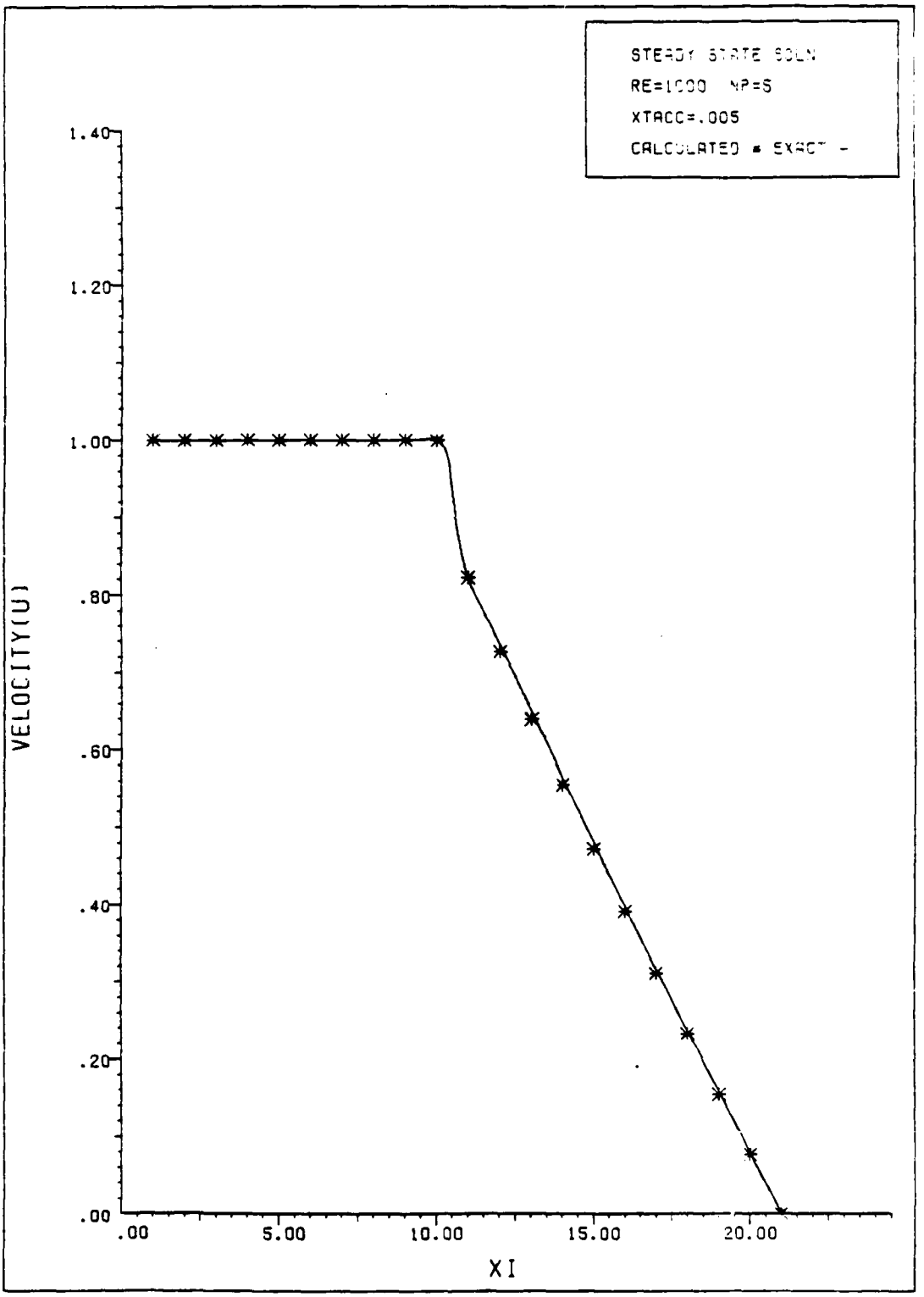
VELOCITY PROFILE FOR CASE 1 IN THE PHYSICAL PLANE

Figure 11



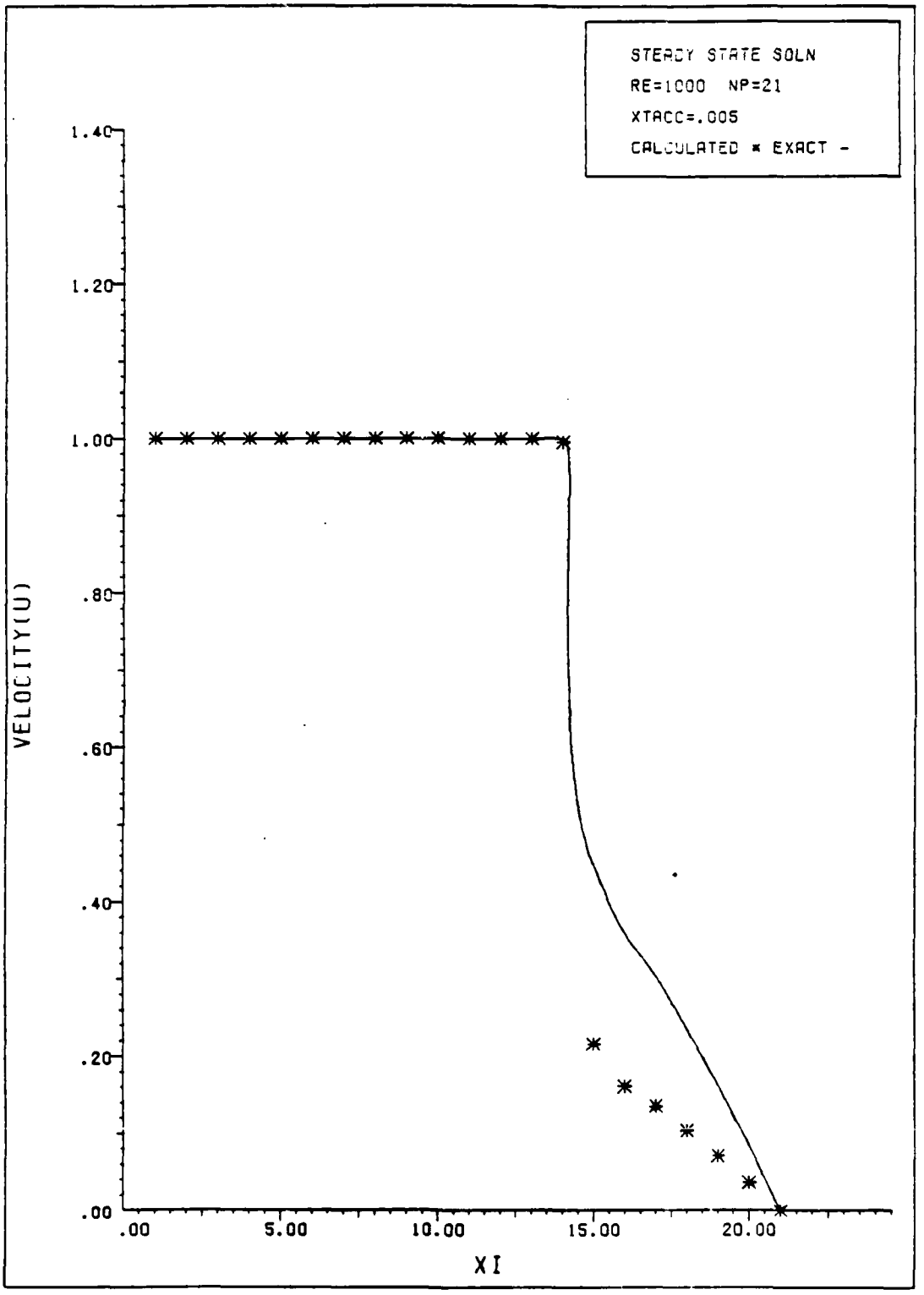
VELOCITY PROFILE FOR CASE 1

Figure 12



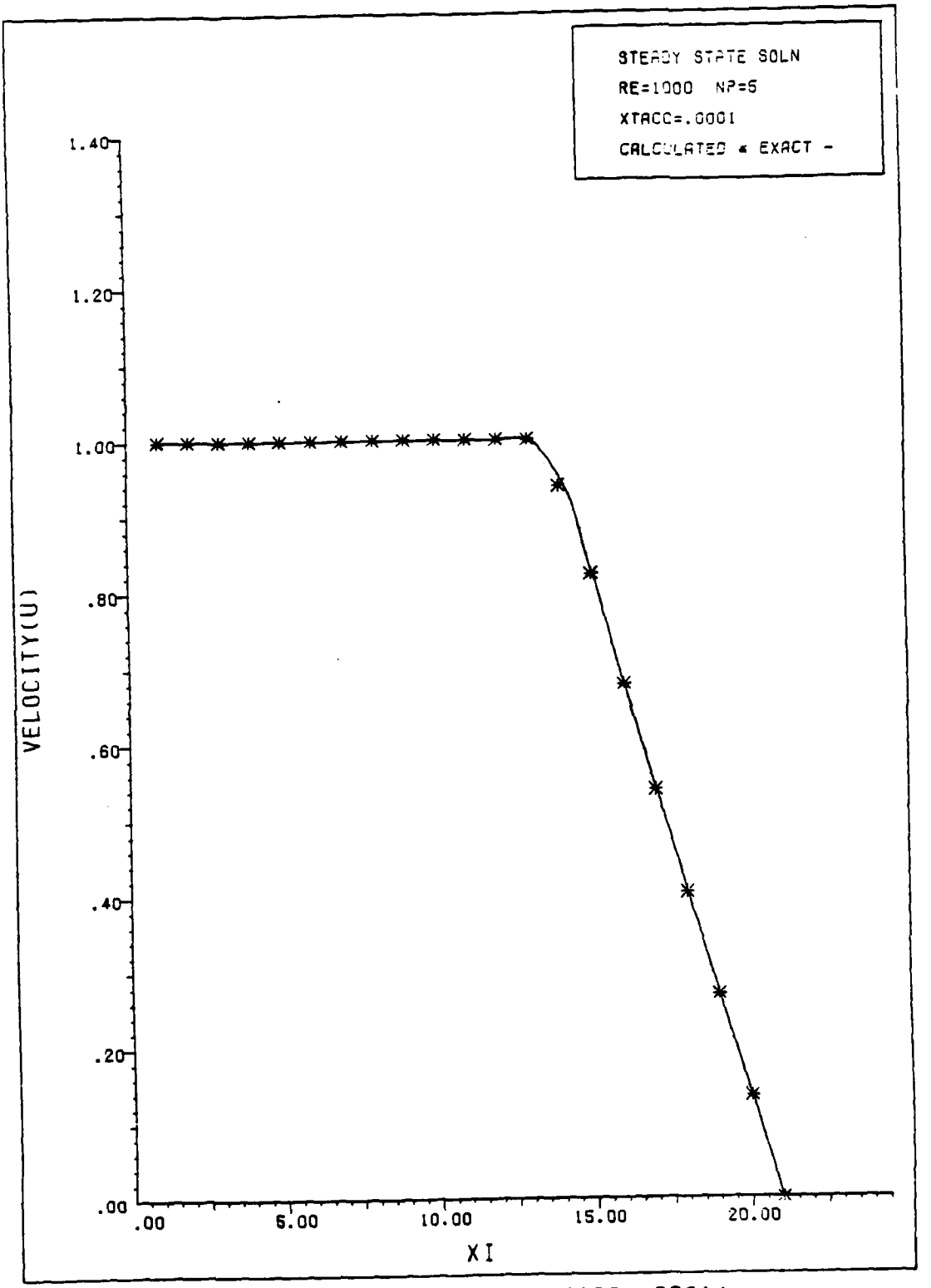
VELOCITY PROFILE FOR CASE 2 (10 PT LINEAR REGION)

Figure 13



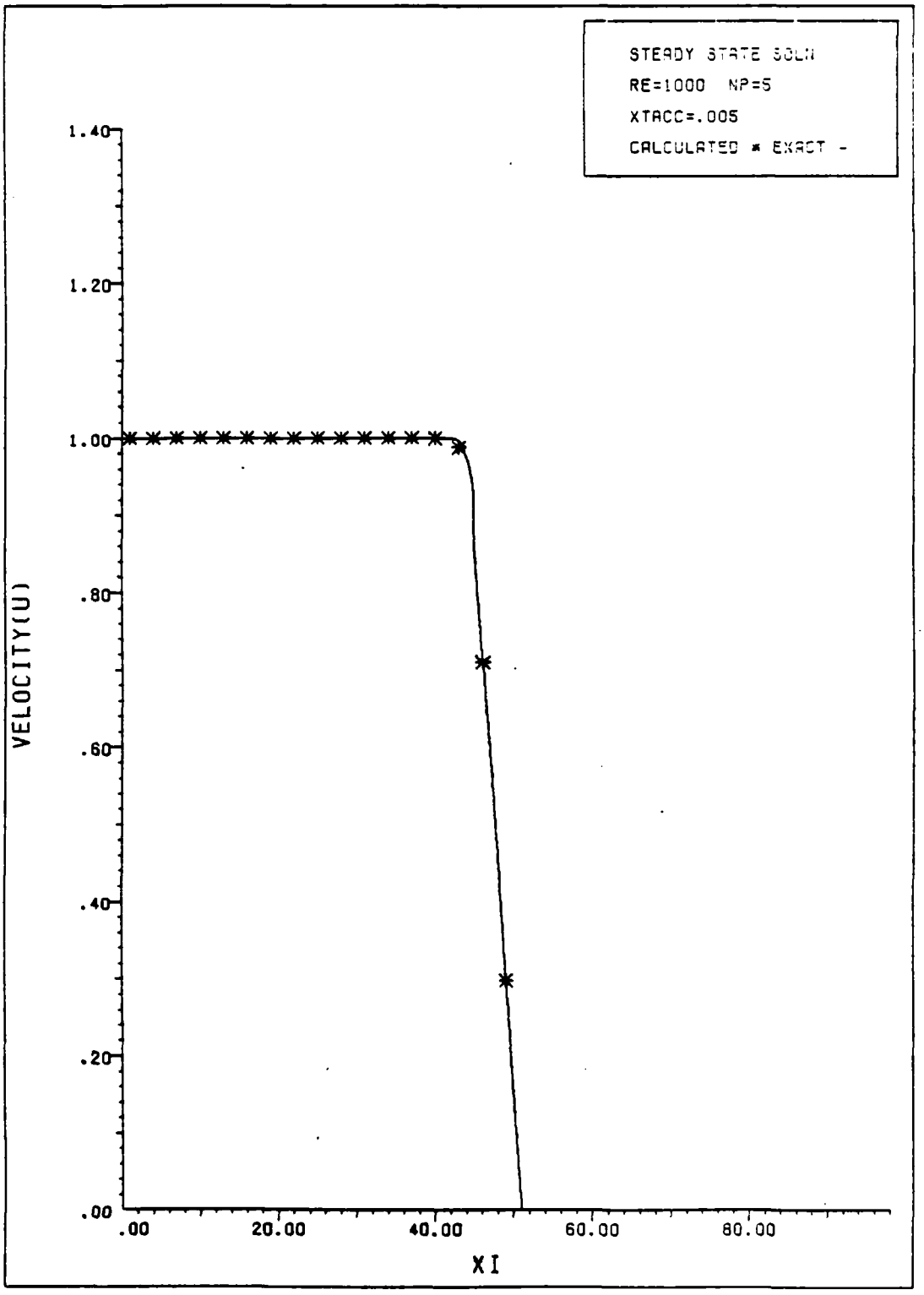
VELOCITY PROFILE FOR CASE 3 (NP=21)

Figure 14



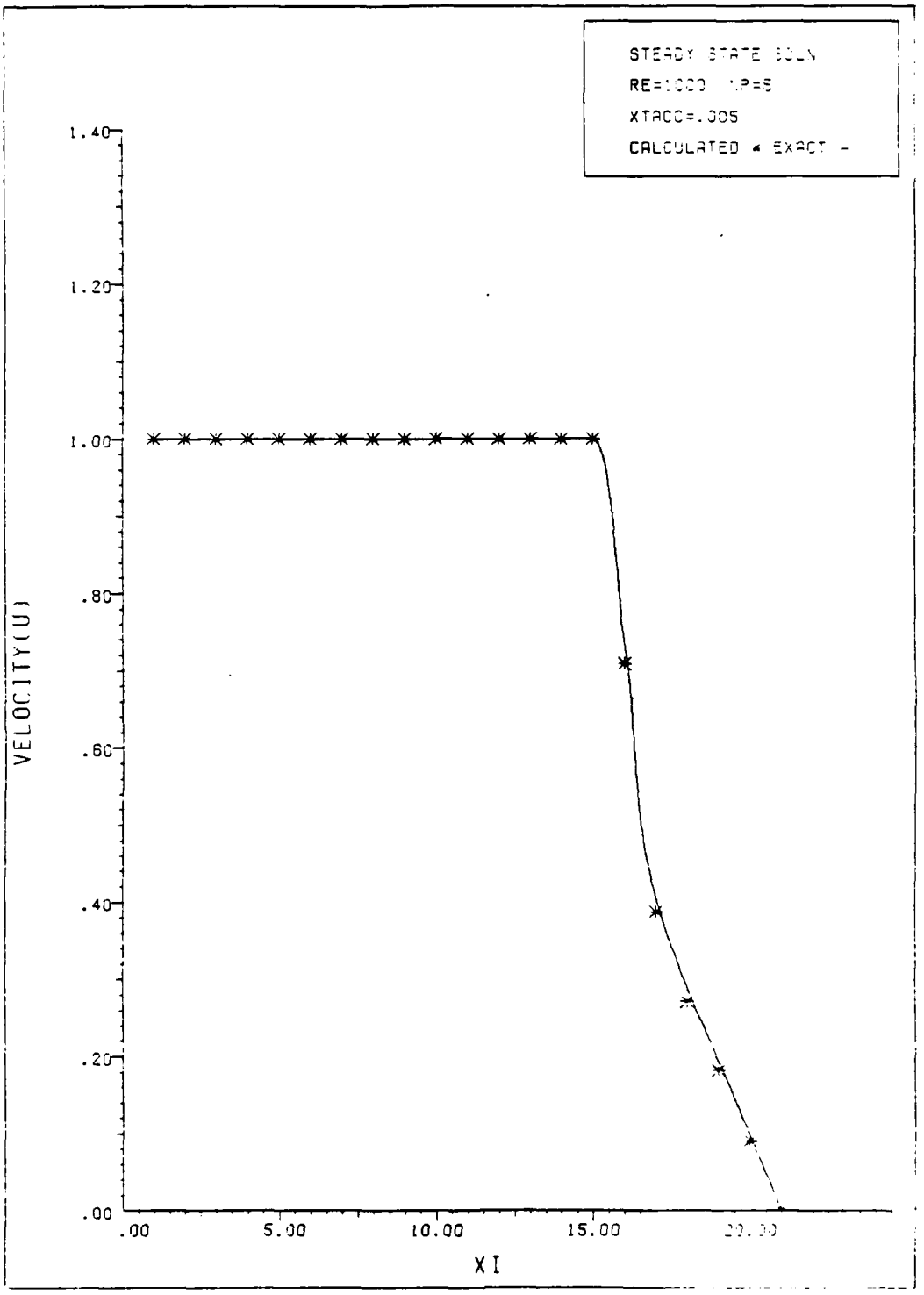
VELOCITY PROFILE FOR CASE 4 (XTACC=.0001)

Figure 15



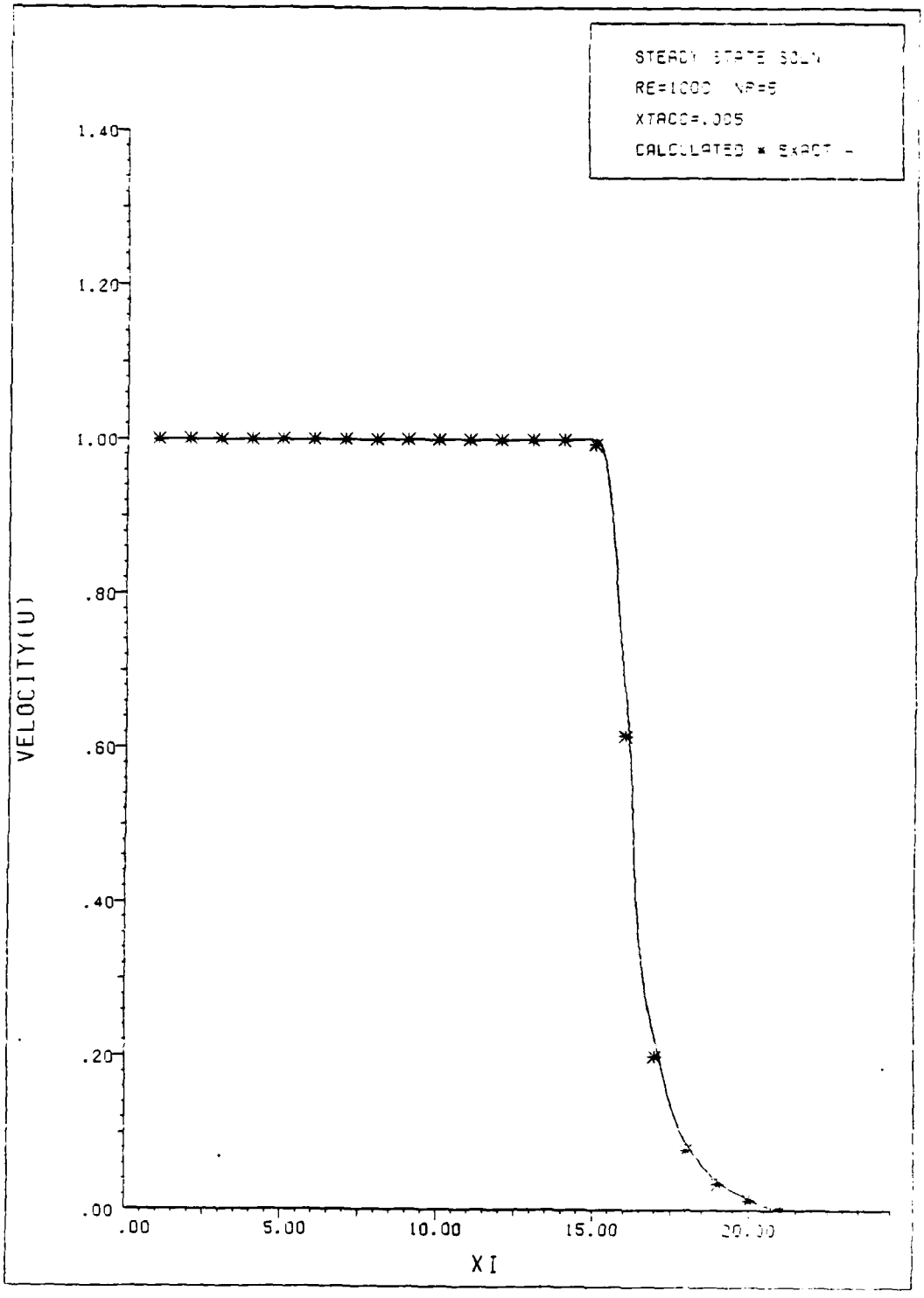
VELOCITY PROFILE FOR CASE 5 (51 GRID POINTS)

Figure 16



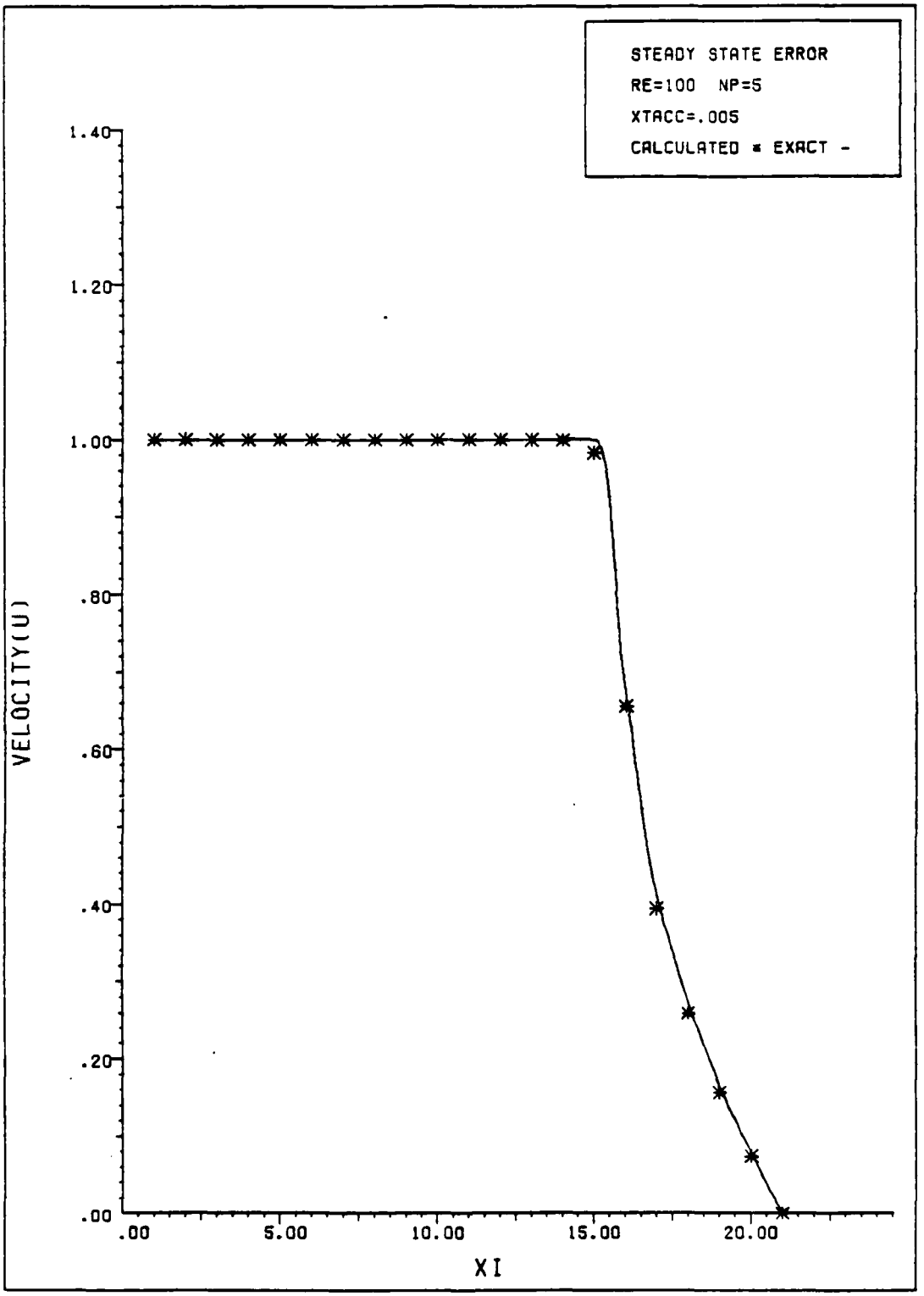
VELOCITY PROFILE FOR CASE 6 (SLOPE ≥ .01)

Figure 17



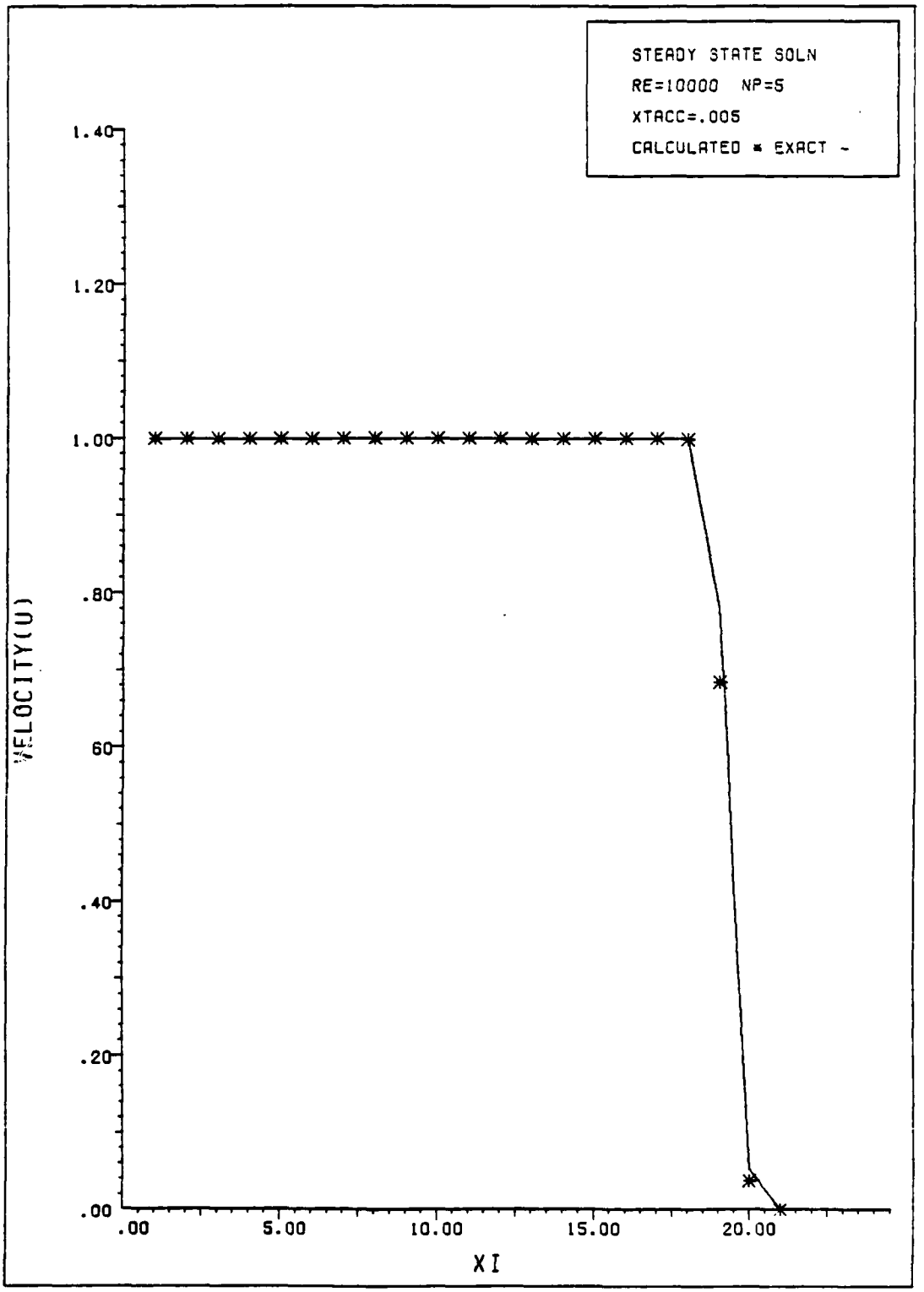
VELOCITY PROFILE FOR CASE 7 (P ≥ 0)

Figure 18



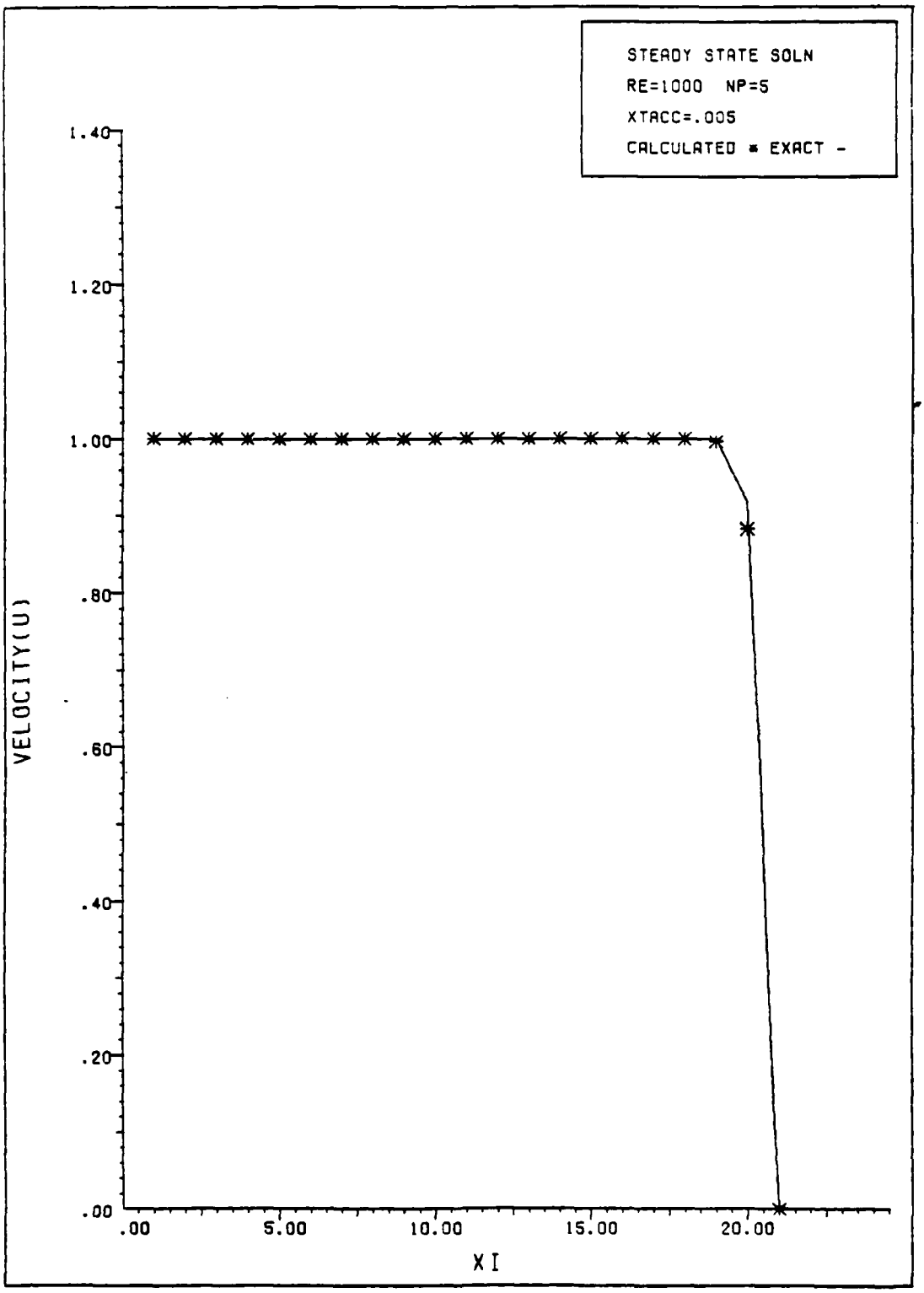
VELOCITY PROFILE FOR CASE 8 (RE=100)

Figure 19



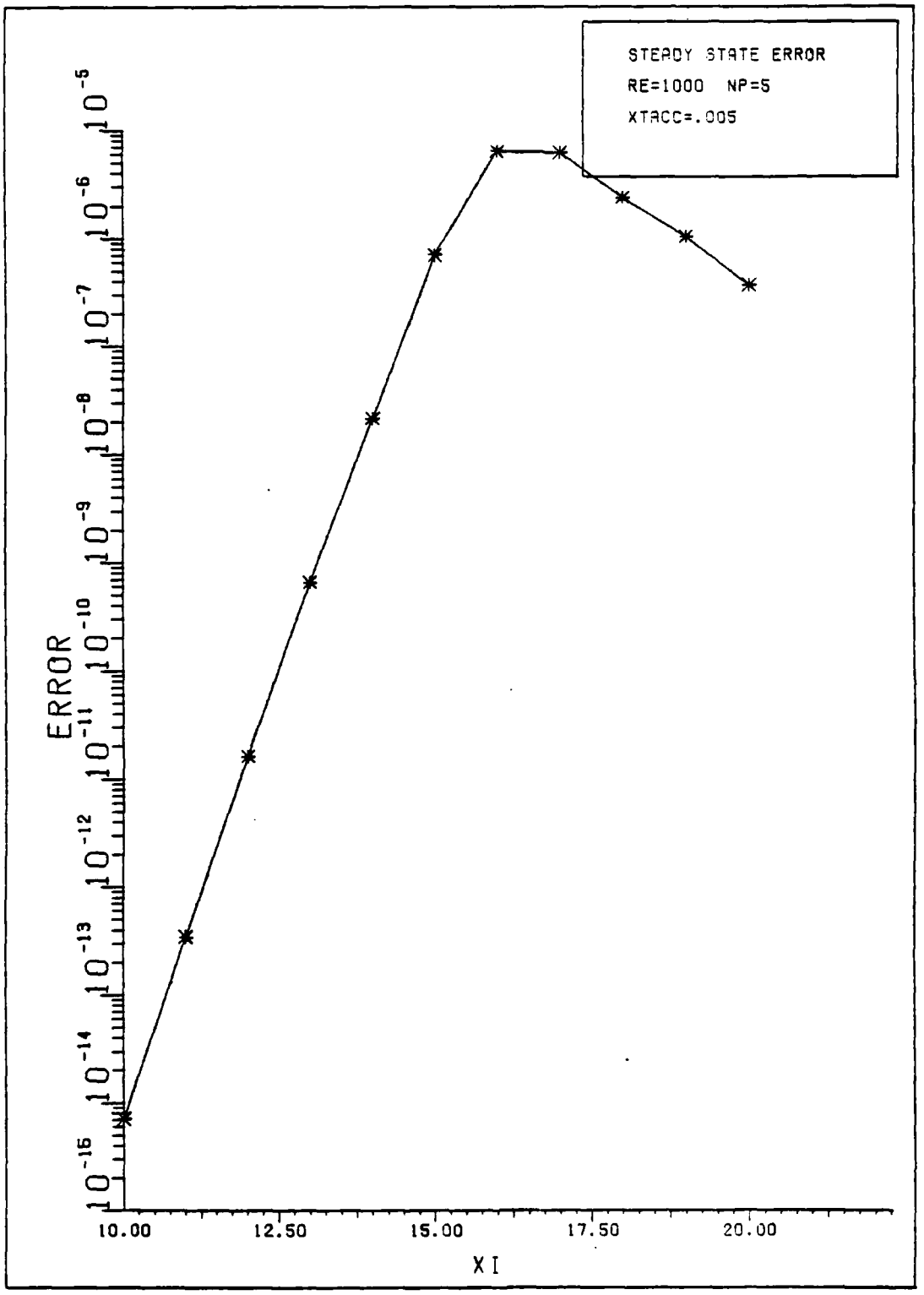
VELOCITY PROFILE FOR CASE 9 (RE=10000)

Figure 20



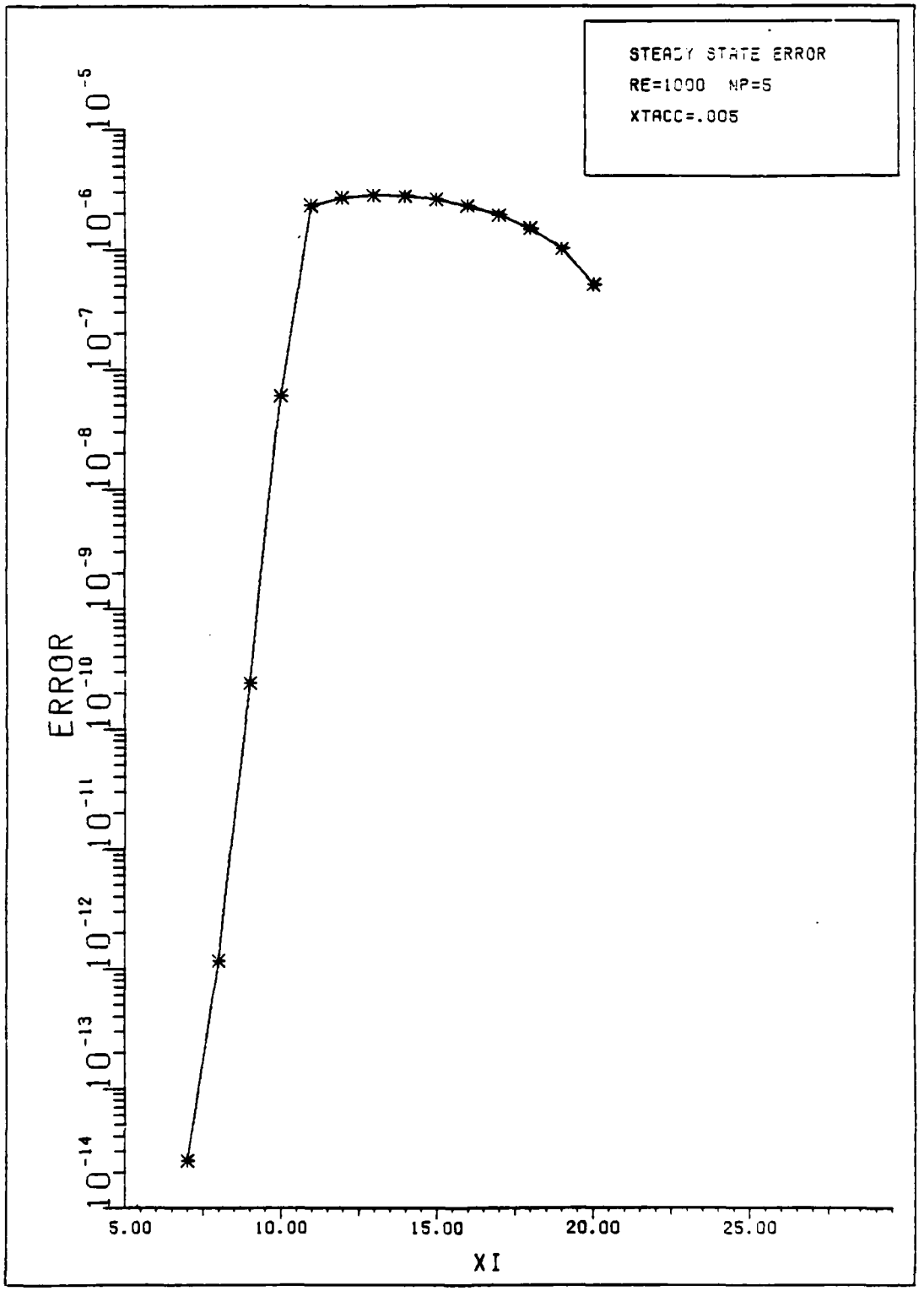
VELOCITY PROFILE FOR CASE 10

Figure 21



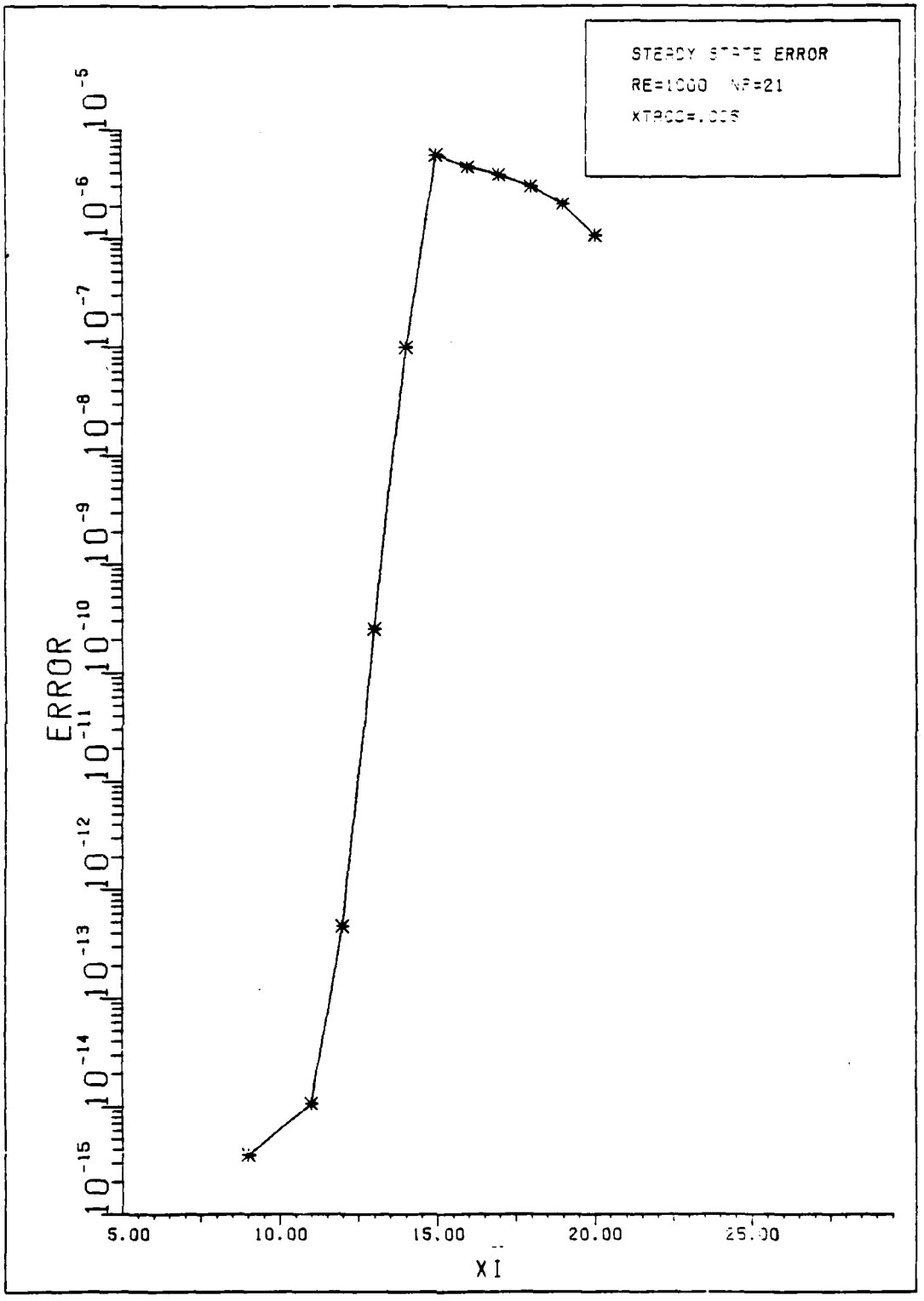
ERROR BETWEEN TIME STEPS FOR CASE 1

Figure 22



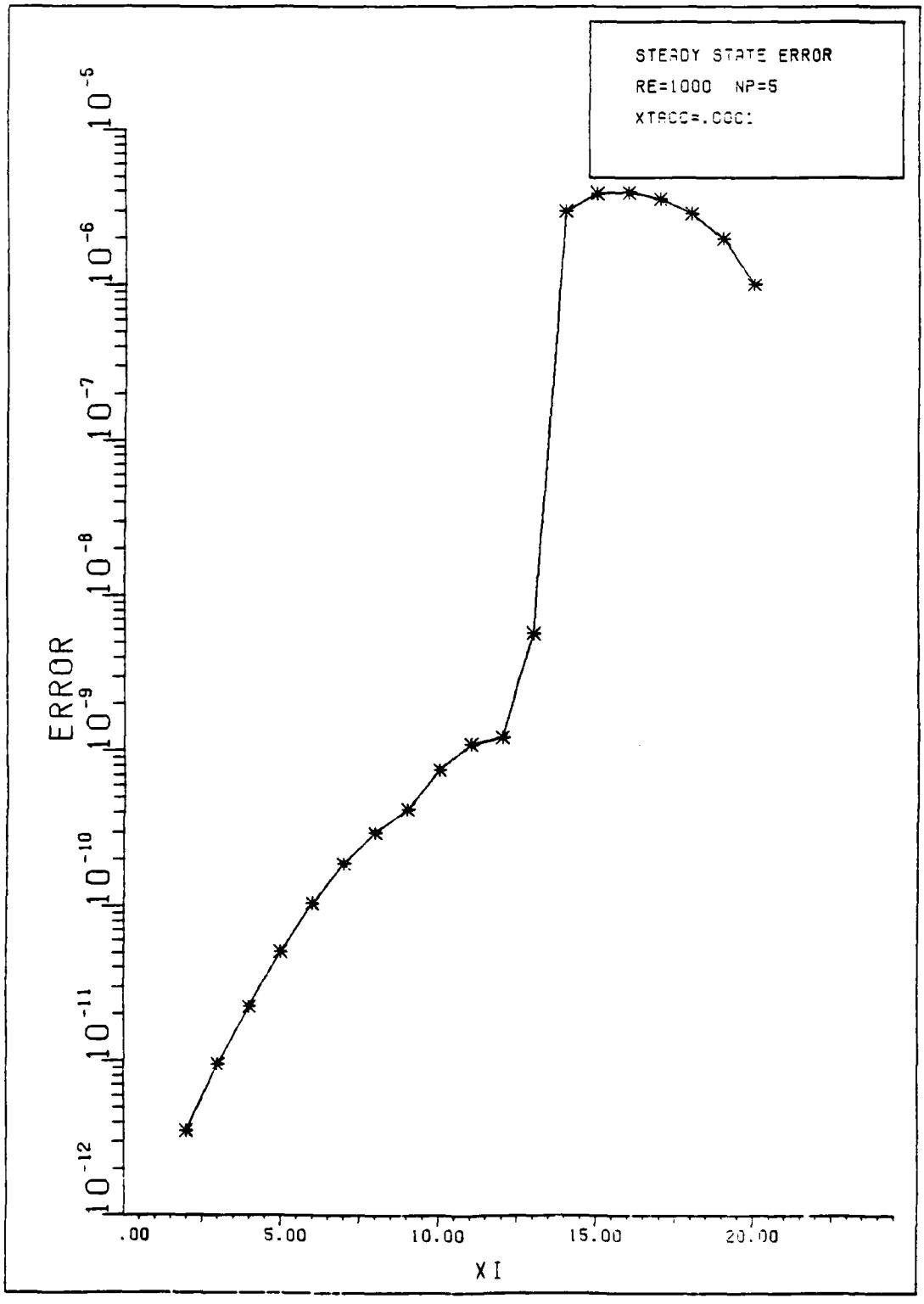
ERROR BETWEEN TIME STEPS FOR CASE 2

Figure 23



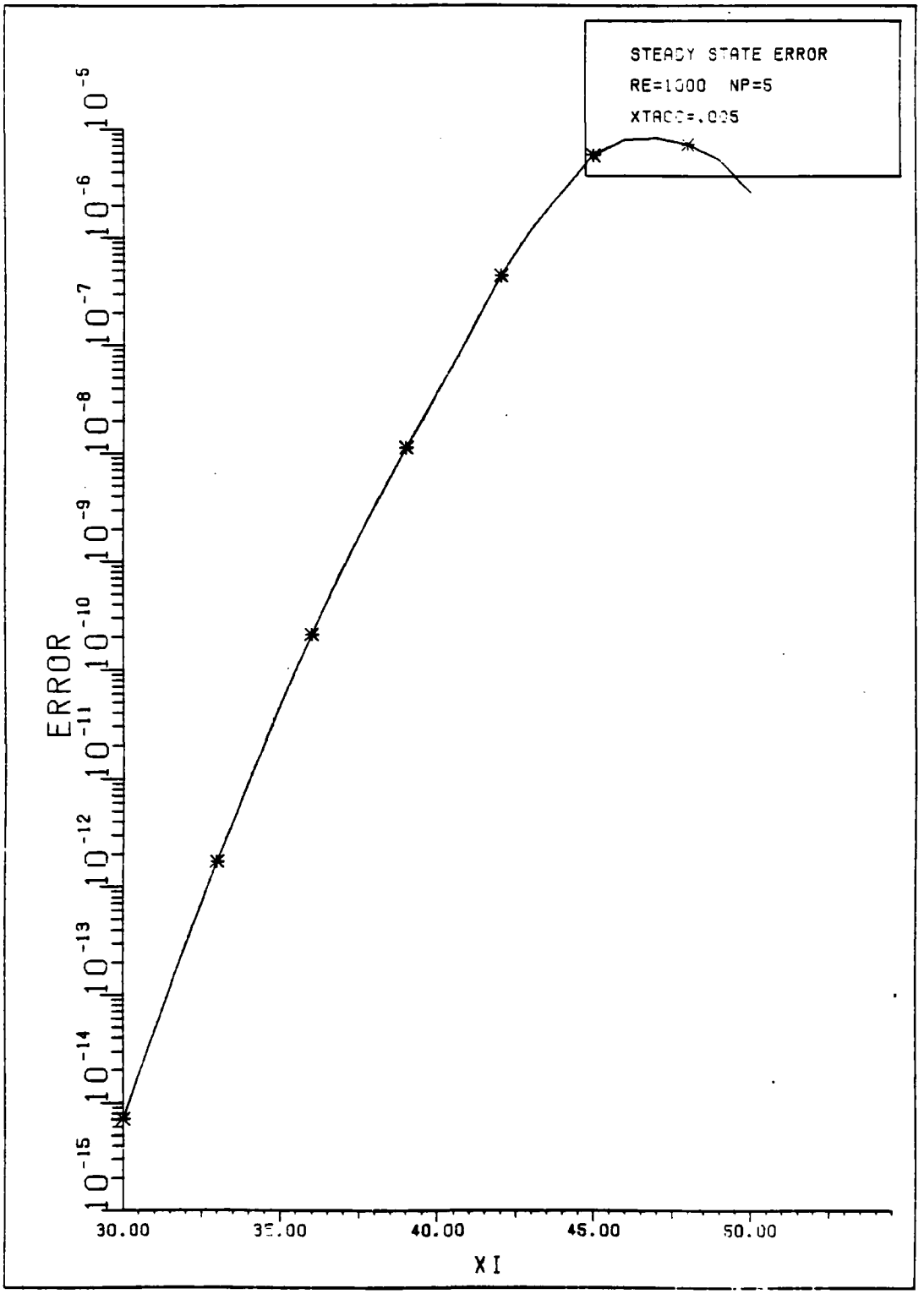
ERROR BETWEEN TIME STEPS FOR CASE 3 (NP=21)

Figure 24



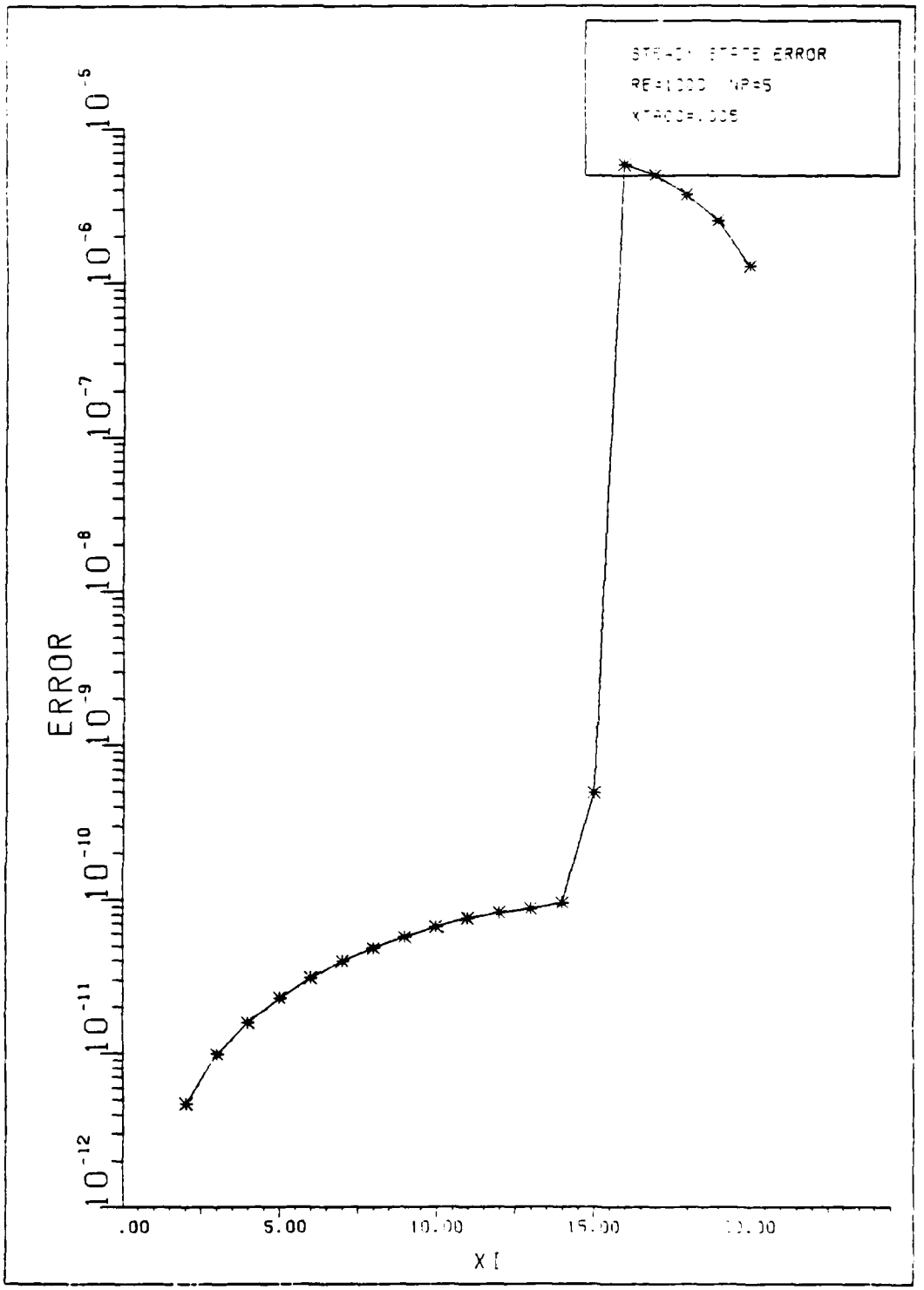
ERROR BETWEEN TIME STEPS FOR CASE 4 (XTACC=.0001)

Figure 25



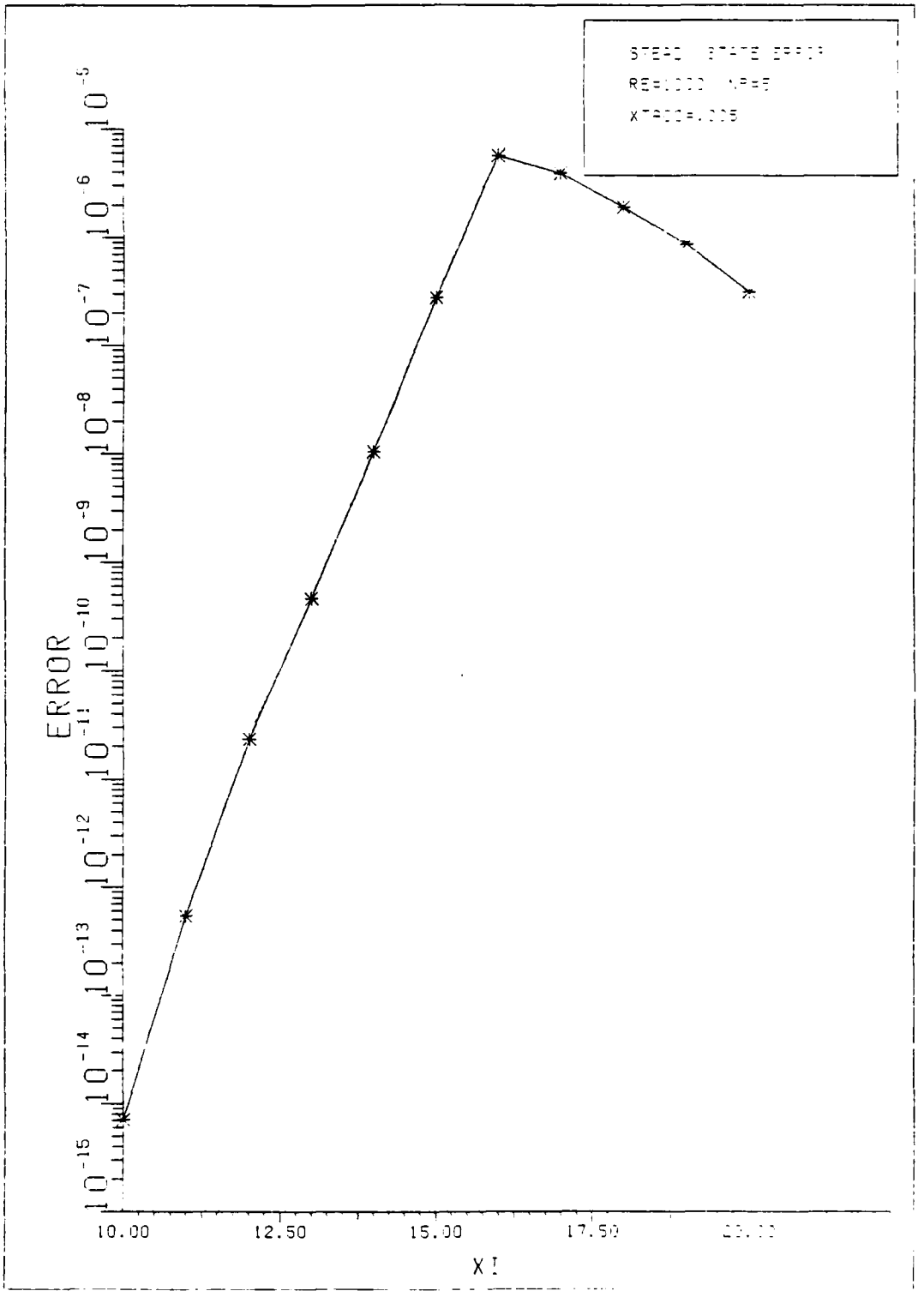
ERROR BETWEEN TIME STEPS FOR CASE 5 (51 GRID PTS)

Figure 26



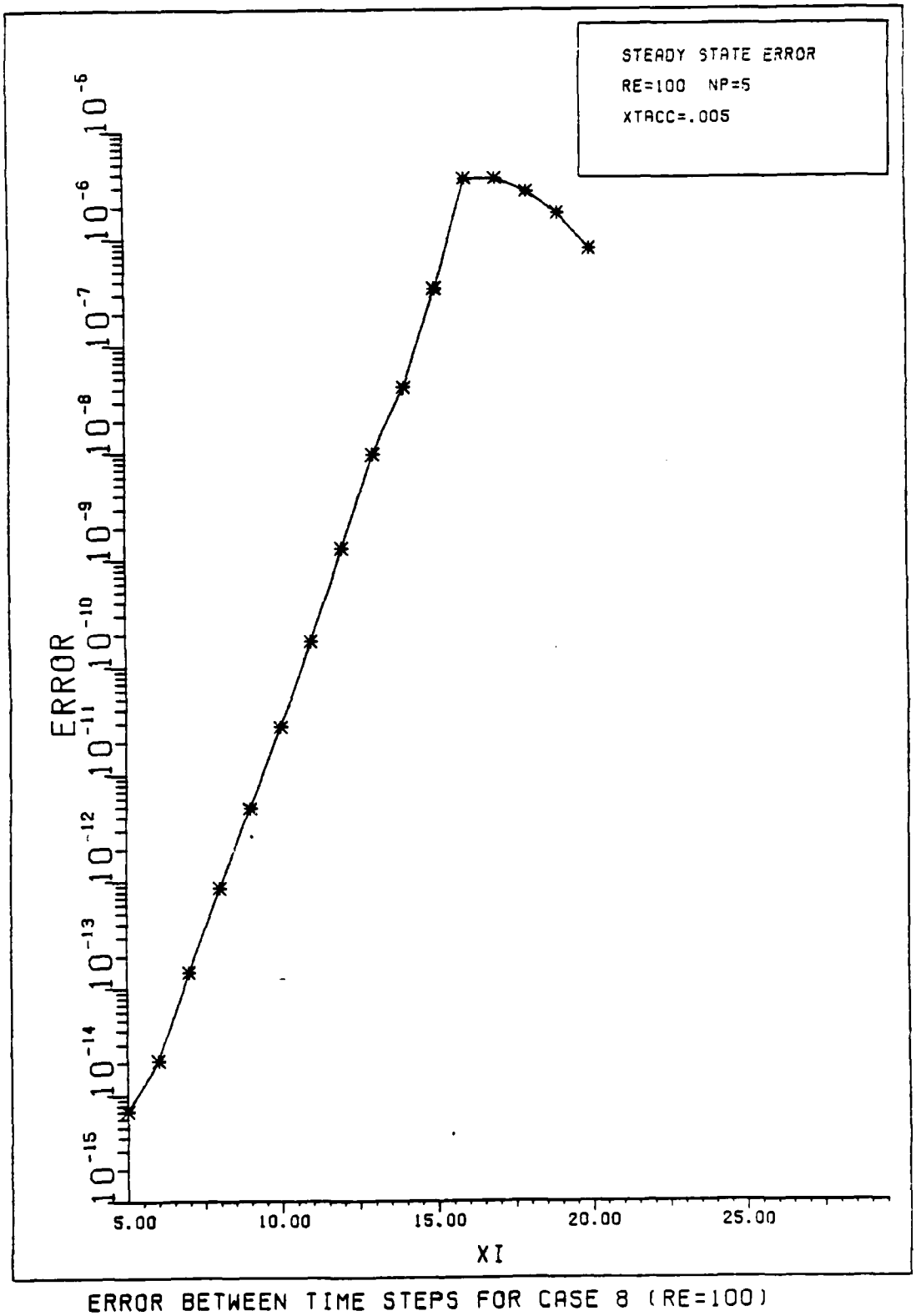
ERROR BETWEEN TIME STEPS FOR CASE 6 (SLOPE = .01)

Figure 27



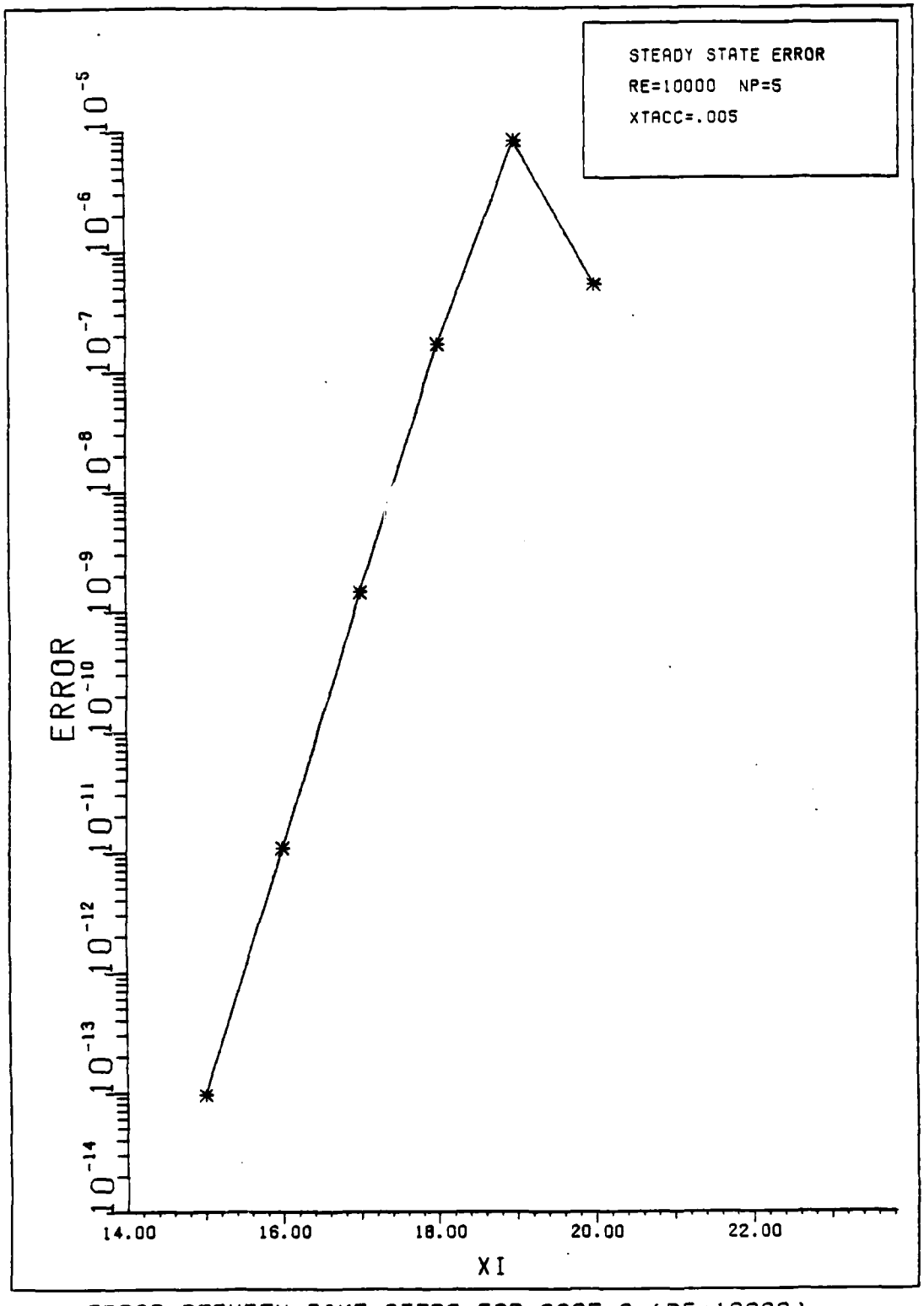
ERROR BETWEEN TIME STEPS FOR CASE

Figure 28



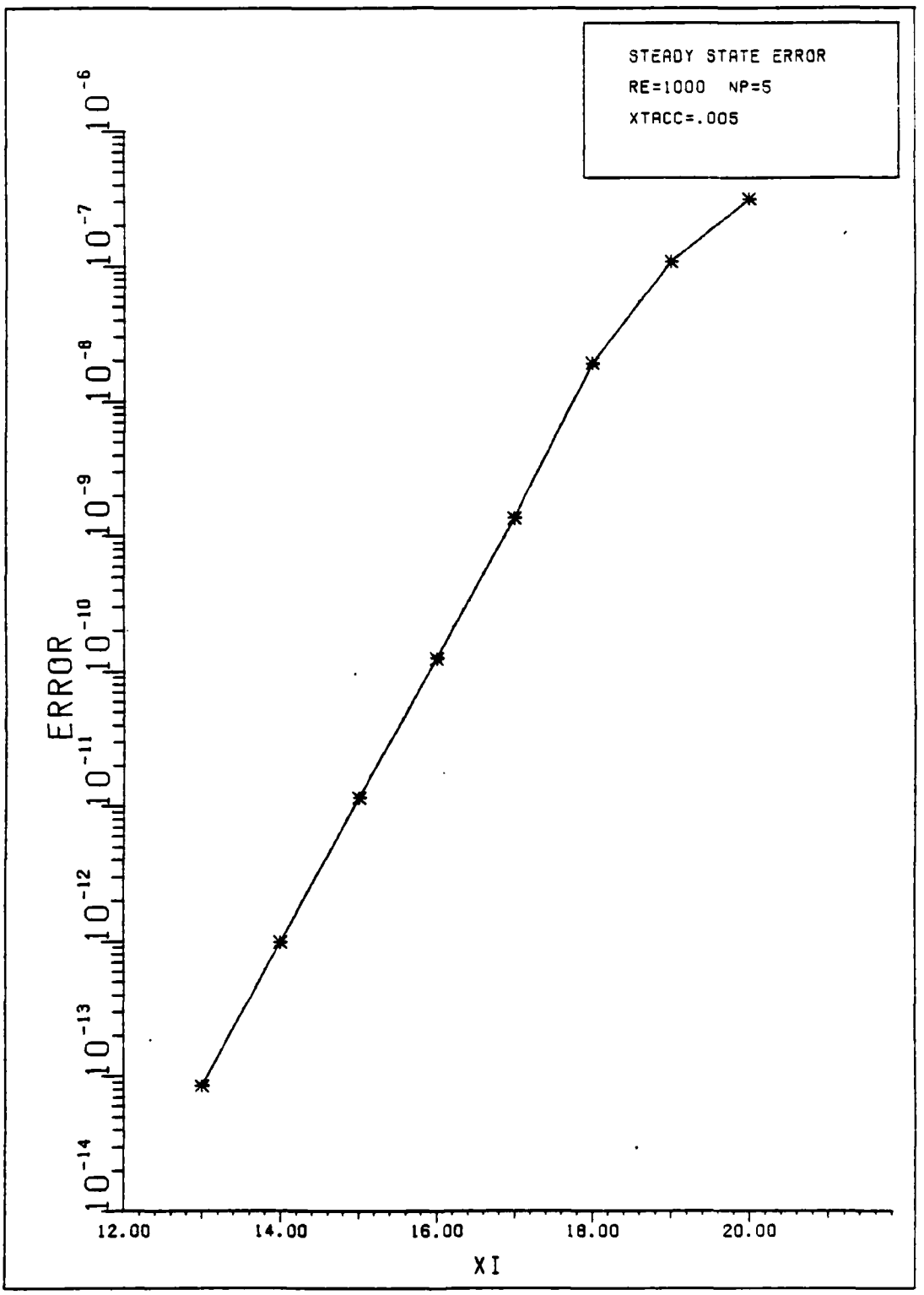
ERROR BETWEEN TIME STEPS FOR CASE 8 (RE=100)

Figure 29



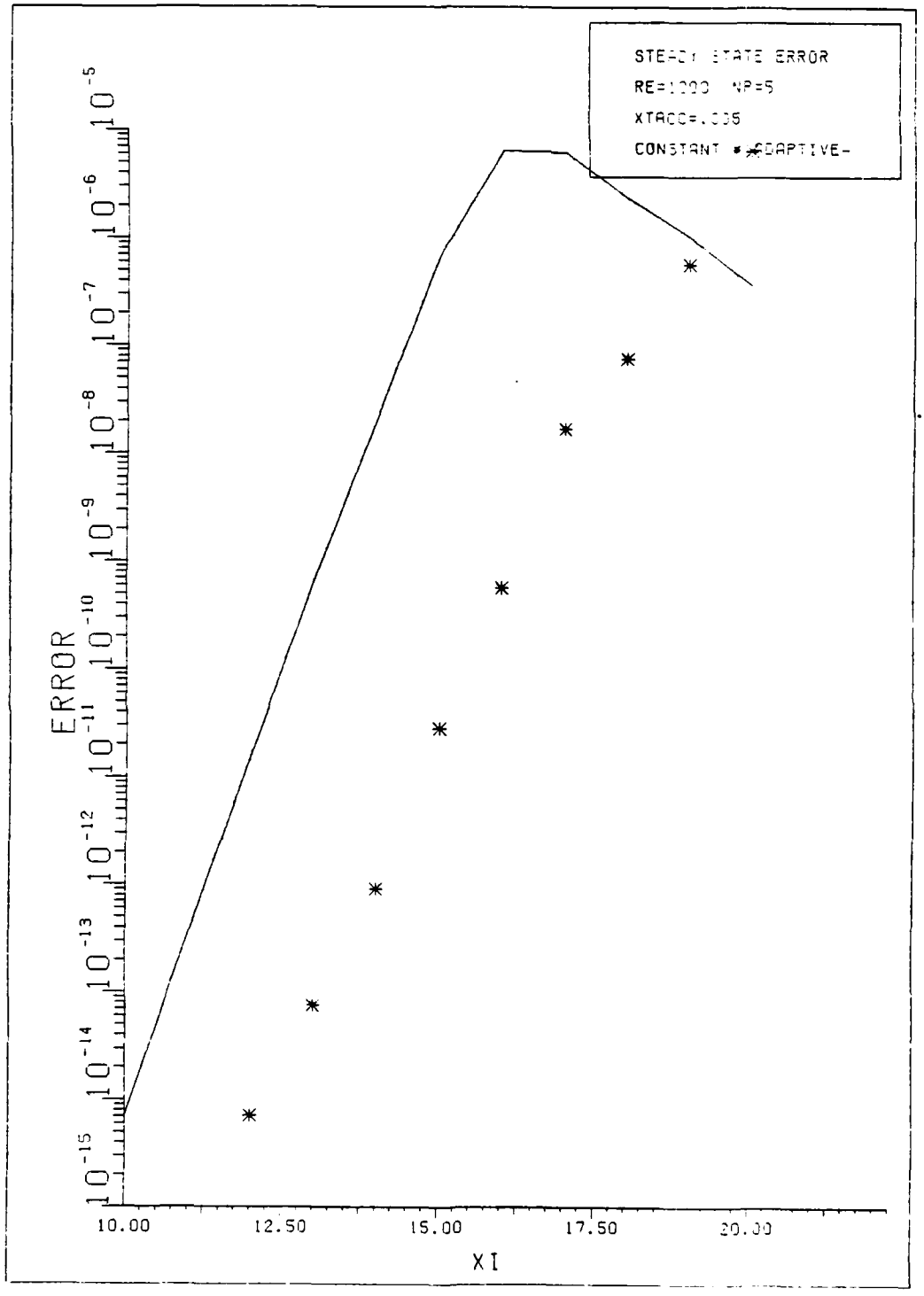
ERROR BETWEEN TIME STEPS FOR CASE 9 (RE=10000)

Figure 30



ERROR BETWEEN TIME STEPS FOR CASE 10

Figure 31



ERROR COMPARISON, CONSTANT AND ADAPTIVE GRIDS

Figure 32

Appendix B: Computer Program BURG Listing

```
      PROGRAM BURG

C*************************************************************
C     UNSTEADY SOLUTION OF THE 1-D BURGER'S EQUATION BY AN    *
C     OPTIMIZED SOR METHOD COUPLED WITH A GRID OPTIMIZATION   *
C     ROUTINE BASED ON A TRUNCATION ERROR ANALYSIS.  THE      *
C     PROCEDURE IS  NOT SELF-STARTING, THUS REQUIRING         *
C     AN INITIAL GUESS FOR THE GRID GENERATION CONTROL        *
C     PARAMETER, P.                                           *
C*************************************************************

      COMMON /A/IMAX,U(51),ZI(51),X(51),P(51),DX(51),RE
      COMMON /B/KGRID,GRIDACC,XMIN,XMAX
      COMMON /C/AAA(51),CCC(51),DDD(51),GGG(51),WWW(51)
      REAL DXX(51),XT(51),XN(51),UN(51),E(51),G(51),C(3),XIN(51)
      REAL LBC,RBC,XMIN,XMAX,ALPHA,BETA,GAMMA,DX2,RE,A,B,USTAR,W,
     # UOLD,SUM,ERRMAX,ERRAVE,DIFF(51),GRIDACC,SOLNACC,DU,DELT,T
      INTEGER I,K,N,NT

CC********* READ INPUT DATA ***********************************
C                                                             *
C     TYPICAL VALUES FOR INPUT CONSTANTS                      *
C-------------------------------------------------------------*
C                                                             *
C     CONSTANT        TYPICAL VALUE                           *
C     --------        -------------                           *
C     IMAX            21                                      *
C     NP              5                                       *
C     NTERMS          3                                       *
C     RE              1000                                    *
C     XTACC           .005                                    *
C*************************************************************

      READ(7,*) IMAX,KGRID,KSOLN,LBC,RBC,XMIN,XMAX,NP,NTERMS
      READ(7,*) RE,GRIDACC,SOLNACC,DELT,NT,XTACC
      READ(7,*)(P(I),I=1,IMAX)
      CALL DATE(ADATE)
      CALL TIME(ATIME)
      WRITE(6,45)  ADATE,ATIME
      WRITE(6,50) IMAX,KSOLN,LBC,XMIN,RBC,XMAX,RE,GRIDACC,SOLNACC,
     #NTERMS,NP
      WRITE(6,51)  DELT,NT,KGRID,XTACC

CC********* SET INITIAL CONDITIONS ****************************

      T=0.0
      N=0
      DO 10 I=1,IMAX
        ZI(I)=FLOAT(I)
        X(I)=(-EXP(-P(I)*I)+EXP(-IMAX*P(I)))/(EXP(-P(I))
     #  -EXP(-IMAX*P(I)))
        XIN(I)=X(I)
        U(I)=1.0
 10   CONTINUE

CC********* SET BOUNDARY CONDITIONS ***************************

      U(1)=-TANH(RE*X(1)*0.5)
      U(IMAX)=-TANH(RE*X(IMAX)*0.5)
      XT(1)=0.0
      XT(IMAX)=0.0


CC******** CALCULATE SOLUTION FOR EACH TIME STEP, N ************
```

```
        CALL SECOND(CPI)
 990    N=N+1
        T=T+DELT
        IF (N .GT. NT) GO TO 1000

C****** SHIFT TIME LEVEL **********************************

        DO 105 I=1,IMAX
          XN(I)=X(I)
 105      UN(I)=U(I)
        WRITE(6,85) T

CC******** CALCULATE GRID ***********************************

      IF(N.LE.5)GOTO 24
      IF(ABS(XTAVE).LE.XTACC)GOTO 25
 24   CALL EXPGRD
 25   CONTINUE

CC******** WRITE GRID DATA ***********************************

      XTAVE=0.0
        WRITE(6,52)
        DO 20 I=2,IMAX-1
          DX(I)=0.5*(X(I+1)-X(I-1))
          DXX(I)=X(I+1)-2.0*X(I)+X(I-1)
          XT(I)=(X(I)-XN(I))/DELT
          XTAVE=XTAVE+XT(I)
          WRITE(6,58)I,X(I),DX(I),DXX(I),XT(I),P(I)
 20     CONTINUE
        XTAVE=XTAVE/(IMAX-2)
        WRITE(6,59)XTAVE

CC******** K IS THE SOR ITERATION LOOP INDEX ******************

        K=0
 999    K= K+1
        ERRMAX=0.0
        SUM=0.0

CC******** I IS THE SPACE LOOP INDEX **************************

        DO 110 I=2,IMAX-1
          APLUS=X(I+1)-X(I)
          AMINUS=X(I)-X(I-1)
          ALPHA=1.0/RE/DX(I)*(1.0/APLUS+1.0/AMINUS)
          BETA=(U(I)-XT(I))/DX(I)

C*************************************************************
C       USE A FIRST ORDER DIFFERENCE NEXT TO THE  BOUNDARY     *
C       POINTS AND A SECOND ORDER UPWIND DIFFERENCE INTERIOR   *
C       TO THESE POINTS FOR THE CONVECTIVE TERM                *
C*************************************************************

            IF (BETA .GE.0.0) THEN
              IF (I .EQ. 2) THEN
                DU=-U(I-1)
                GAMMA=ALPHA+BETA+1.0/DELT
                A=1.0/RE/DX(I)/APLUS/GAMMA
                B = (1.0/RE/DX(I)/AMINUS+BETA)/GAMMA
              ELSE
                DU = 0.5*U(I-2) - 2.0*U(I-1)
                GAMMA=ALPHA+1.5*BETA+1.0/DELT
                A=1.0/RE/DX(I)/APLUS/GAMMA
                B=(1.0/RE/DX(I)/AMINUS+2.0*BETA)/GAMMA
```

```
              END IF
            ELSE
              IF (I .EQ. (IMAX-1)) THEN
                DU = U(I+1)
                GAMMA=ALPHA-BETA+1.0/DELT
                A=(1.0/RE/DX(I)/AMINUS-BETA)/GAMMA
                B=1.0/RE/DX(I)/AMINUS/GAMMA
              ELSE
                DU = -0.5*U(I+2) + 2.0*U(I+1)
                GAMMA=ALPHA-1.5*BETA+1.0/DELT
                A=(1.0/RE/DX(I)/APLUS-2.0*BETA)/GAMMA
                B=1.0/RE/DX(I)/AMINUS/GAMMA
              END IF
            END IF

CC******** CALCULATE THE GAUSS-SEIDEL VALUE ********************

          USTAR=(UN(I)/DELT+1.0/RE/DX(I)*(U(I+1)/APLUS+U(I-1)/AMINUS)
     #     -BETA*DU)/GAMMA

CC******** CALCULATE THE OPTIMUM RELAXATION FACTOR *************

            CALL WOPT(A,B,W)
            IF(N .EQ. 1 .AND. K .EQ. 1) W=1.0

C**** CALCULATE SOLUTION & ERROR BETWEEN SOR ITERATIONS ********

            UOLD = U(I)
            U(I) = U(I) + W*(USTAR - U(I))
            DIFF(I) = ABS(U(I) - UOLD)
            SUM = SUM + DIFF(I)
 110        CONTINUE

CC******** CHECK FOR SOR ITERATION CONVERGENCE *****************

          ERRAVE = SUM/IMAX
          IF(K.EQ.KSOLN) GOTO 901
          IF (ERRAVE .GE. SOLNACC) GO TO 999
 901  WRITE(6,79)


C***** CALCULATE NEW GRID GENERATION CONTROL FUNCTION P *********

      IF(N.LE.5)GOTO 34
      IF(ABS(XTAVE).LE.XTACC)GOTO 35
 34   CALL NEWP(C,NP,G,NTERMS)
 35   CONTINUE

CC******** CALCULATE ANALYTIC STEADY STATE SOLUTION ************

        WRITE(6,87)
        ERR1=0.0
        SUM1=0.0
        ERR2=0.0
        SUM2=0.0
        ERR3=0.0
        SUM3=0.0
        DO 120 I=1,IMAX
          E(I)=-TANH(RE*X(I)*0.5)
          DIF1=ABS(U(I)-E(I))
          SUM1=SUM1+DIF1
          DIF2=ABS(U(I)-G(I))
          SUM2=SUM2+DIF2
          DIF3=ABS(G(I)-E(I))
          SUM3=SUM3+DIF3
```

```
 120     CONTINUE
         ERRAV1=SUM1/IMAX
         ERRAV2=SUM2/IMAX
         ERRAV3=SUM3/IMAX

C***** CALCULATE AVERAGE ERROR BETWEEN TIME ITERATIONS *********

       SUM=0.0
         DO 180 I=1,IMAX
           DIFF(I)=ABS(U(I)-UN(I))
           SUM=SUM+DIFF(I)
           WRITE(6,58)I,X(I),U(I),E(I),DIFF(I),Q(I)
 180     CONTINUE
       ERRAVE=SUM/IMAX
       WRITE(13,*)N,ERRAVE
       WRITE(6,80)K,ERRAV1,ERRAV2,ERRAV3
       IF (ERRAVE .GE. SOLNACC) GO TO 990
       DO 190 I=1,IMAX
         WRITE(9,89) ZI(I),U(I),E(I),X(I)
         WRITE(10,*)ZI(I),U(I),E(I)
         WRITE(11,*)ZI(I),X(I),XIN(I)
         WRITE(14,*)ZI(I),DIFF(I)
         WRITE(15,*)X(I),U(I),E(I)
         WRITE(16,*)X(I),DIFF(I)
         WRITE(17,*)ZI(I),DX(I)
         WRITE(18,*)ZI(I),DXX(I)
         WRITE(19,*)X(I),DX(I)
         WRITE(20,*)X(I),DXX(I)
 190   CONTINUE

       CALL SECOND(CPF)
       CPU=CPF-CPI
       WRITE(6,85)CPU
       STOP


  45   FORMAT('1',/,T5,A10,2X,A10)

  50   FORMAT(/,T10,'1-D BURGERS EQ. SOLN USING OPTIMIZED SOR METHOD'
      #,' & OPTIMUM GRID',//,T25,'**********INPUT DATA**********',
      #/,5X,'# OF GRID POINTS =',I3,5X,'# OF ITERATIONS ALLOWED'
      #,' =',I4,/,5X,'L.B.C. =',F8.5,'  AT X = ',F8.5,/5X,
      #'R.B.C. =',F8.5,'  AT X = ',F8.5,/,5X,'REYNOLDS # =',
      #F9.0,/,5X,'GRID ACC. CK. =',E9.3,3X,'SOLN ACC CK. =',E9.3,/,5X,
      #'#OF TERMS IN LEAST SQUARES APPROX. =',I3,3X,'NP=',I3)

  51   FORMAT(5X,'DELTA T=',F10.5,5X,'# OF TIME STEPS =',I5,/,
      #5X,'GRID ITERATIONS ALLOWED=',I5,/,5X,'XTACC=',F12.7,/)

  52   FORMAT(/,T25,'GRID CALCULATION',/,T4,'ZI',T14,'X',T25,
      #'DX',T38,'DXX',T50,'XT',T62,'P',/)

  58   FORMAT(I5,5(E12.5))

  59   FORMAT(/,T5,'AVERAGE VALUE OF GRID SPEED=',F12.7)

  85   FORMAT(//,T30,'TIME =',F7.3,/)

  79   FORMAT(//)

  80   FORMAT(T5,'SOR ITERATION #',I5,5X,'AVERAGE ERROR ANALYSIS',/,
      #T30,'COMPUTED & ANALYTICAL :',F13.8,/,T30,'COMPUTED & FIT :',
      #8X,F12.8,/,T30,'ANALYTICAL & FIT :',5X,F13.8)

  87   FORMAT(/,T30,'*****  ANALYTIC SOLUTION  *****',//
```

```
     #Y4,'ZI',Y14,'X',Y25,'U',Y35,'U EXACT',Y47,'DIFF',Y60,'FIT',/)

 95   FORMAT(/,T5,'TOTAL CPU TIME :', E12.5)

 98   FORMAT(/,T5,'MAX ITERATIONS EXCEEDED!!!!!',/)

 99   FORMAT(1X,4F10.7)

1000  WRITE(6,98)
      CALL SECOND(CPF)
      CPU=CPF-CPI
      WRITE(6,95)CPU

      STOP
      END



      SUBROUTINE WOPT(A,B,W)

      PJ=2.0*SQRT(A*B)*0.707
      W=2.0/(1.0+SQRT(1.0-PJ**2))
      RETURN
      END



      SUBROUTINE NEWP(C,NP,G,NTERMS)

C*****************************************************************
C     THIS SUBROUTINE CACULATES THE NEW GRID GENERATION CONTROL  *
C     FUNCTION P.   IT IS BASED ON A GLOBAL TRUNCATION ERROR     *
C     ANALYSIS FOR THE FLOW SOLUTION IN THE TRANSFORMED PLANE    *
C*****************************************************************

      COMMON /A/IMAX,U(51),ZI(51),X(51),P(51),DX(51),RE
      REAL C(NTERMS),ZZ(51),UU(51),G(51)
      NN=(NP-1)/2
      WRITE(6,15)
      DO 7 J=1,NP
        ZZ(J)=ZI(J)
        UU(J)=U(J)
 7    CONTINUE
      DO 5 I=2,NN+1
        IF((I.GE.(IMAX-5)).AND.(I.LE.(IMAX-1))) THEN
          NTERMS=2
          CALL LSTSQ(NP,ZZ,UU,C,NTERMS)
          A2=0.0
        ELSE
          NTERMS=3
          CALL LSTSQ(NP,ZZ,UU,C,NTERMS)
          A2=C(3)
        ENDIF
        G(I)=0.0
        DO 9 J=1,NTERMS
          G(I)=G(I)+C(J)*I**(J-1)
 9      CONTINUE

C*****************************************************************
C     TO BE CONSISTENT, DU SHOULD BE UPWIND DIFFERENCED BASED    *
C     ON THE SIGN OF BETA                                        *
C*****************************************************************
```

```
      DU=.5*(U(I+1)-U(I-1))
      DUU=U(I+1)-2.*U(I)+U(I-1)
      DUSIGN=SIGN(1.0,DU)
      IF(ABS(DU).LE.0.1) DU=.1*DUSIGN
      POLD=P(I)
      P(I)=POLD+(DUU-2.*A2)/DU
      IF(P(I).GE.50.0) P(I)=50.0
      IF(P(I).LE.-50.0) P(I)=-50.0
      WRITE(6,16)I,DU,DUU,DUU/DU,POLD,P(I),A2,C(2),C(1)
      WRITE(21,*)ZI(I),DU
      WRITE(22,*)ZI(I),DUU
      WRITE(23,*)X(I),DU
      WRITE(24,*)X(I),DUU
5     CONTINUE

      DO 10 I=2+NN,IMAX-NN-1
        DO 20 J=1,NP
          ZZ(J)=ZI(I+J-NN-1)
          UU(J)=U(I+J-NN-1)
20      CONTINUE
        IF((I.GE.(IMAX-5)).AND.(I.LE.(IMAX-1))) THEN
          NTERMS=2
          CALL LSTSQ(NP,ZZ,UU,C,NTERMS)
          A2=0.0
        ELSE
          NTERMS=3
          CALL LSTSQ(NP,ZZ,UU,C,NTERMS)
          A2=C(3)
        ENDIF
        G(I)=0.0
        DO 11 J=1,NTERMS
          G(I)=G(I)+C(J)*I**(J-1)
11      CONTINUE
        DU=.5*(U(I+1)-U(I-1))
        DUU=U(I+1)-2.*U(I)+U(I-1)
        DUSIGN=SIGN(1.0,DU)
        IF(ABS(DU).LE.0.1) DU=.1*DUSIGN
        POLD=P(I)
        P(I)=POLD+(DUU-2.*A2)/DU
        IF(P(I).GE.50.0) P(I)=50.0
        IF(P(I).LE.-50.0) P(I)=-50.0
        WRITE(6,16)I,DU,DUU,DUU/DU,POLD,P(I),A2,C(2),C(1)
        WRITE(21,*)ZI(I),DU
        WRITE(22,*)ZI(I),DUU
        WRITE(23,*)X(I),DU
        WRITE(24,*)X(I),DUU
10    CONTINUE

      DO 30 J=NP,1,-1
        ZZ(J)=ZI(IMAX+J-NP)
        UU(J)=U(IMAX+J-NP)
30    CONTINUE
      DO 25 I=IMAX-NN,IMAX-1
        IF((I.GE.(IMAX-5)).AND.(I.LE.(IMAX-1))) THEN
          NTERMS=2
          CALL LSTSQ(NP,ZZ,UU,C,NTERMS)
          A2=0.0
        ELSE
          NTERMS=3
          CALL LSTSQ(NP,ZZ,UU,C,NTERMS)
          A2=C(3)
        ENDIF
        G(I)=0.0
        DO 28 J=1,NTERMS
```

```
          G(I)=G(I)+C(J)*I**(J-1)
 26       CONTINUE
          DU= .5*(U(I+1)-U(I-1))
          DUU=U(I+1)-2.*U(I)+U(I-1)
          DUSIGN=SIGN(1.0,DU)
          IF(ABS(DU).LE.0.1) DU=.1*DUSIGN
          POLD=P(I)
          P(I)=POLD+(DUU-2.*A2)/DU
          IF(P(I).GE.50.0) P(I)=50.0
          IF(P(I).LE.-50.0) P(I)=-50.0
          WRITE(8,16)I,DU,DUU,DUU/DU,POLD,P(I),A2,C(2),C(1)
          WRITE(21,*)ZI(I),DU
          WRITE(22,*)ZI(I),DUU
          WRITE(23,*)X(I),DU
          WRITE(24,*)X(I),DUU
 25     CONTINUE
        P(1)=0.
        G(1)=U(1)
        G(IMAX)=U(IMAX)
        P(IMAX)=0.

 15     FORMAT(/,T25,'*********NEW P CALCULATION******',//
       #T4,'ZI',T14,'DU',T28,'DUU',T38,'DUU/DU',
       #T48,'POLD',T60,'PNEW',T74,'A2',/)

 16     FORMAT(I5,8(E12.5))

        RETURN
        END



        SUBROUTINE LSTSQ(NP,ZZ,UU,D,NTERMS)

C***************************************************
C   THIS SUBROUTINE MINIMIZES A SUM OF POLYNOMIALS   *
C   OF DEGREE TWO AND LESS AND CALCULATES THE OPTIMUM *
C   COEFFICIENTS                                     *
C***************************************************

        REAL ZZ(NP),UU(NP),ERROR(21),PJM1(21),PJ(21),S(20),
       #B(20),D(NTERMS),C(20)
        DO 10 J=1,NTERMS
          B(J)=0.
          D(J)=0.
          S(J)=0.
 10     CONTINUE
        C(1)=0.
        DO 20 I=1,NP
          D(1)=D(1)+UU(I)
          B(1)=B(1)+ZZ(I)
          S(1)=S(1)+1.0
 20     CONTINUE
        D(1)=D(1)/S(1)
        DO 30 I=1,NP
          ERROR(I)=UU(I)-D(1)
 30     CONTINUE
        IF(NTERMS.EQ.1) RETURN
        B(1)=B(1)/S(1)
        DO 40 I=1,NP
          PJM1(I)=1.
          PJ(I)=ZZ(I)-B(1)
 40     CONTINUE
        J=1
```

```
 41   J=J+1
      DO 50 I=1,NP
        P=PJ(I)
        D(J)=D(J)+ERROR(I)*P
        P=P*PJ(I)
        B(J)=B(J)+ZZ(I)*P
        S(J)=S(J)+P
 50   CONTINUE
      D(J)=D(J)/S(J)
      IF(J.EQ.NTERMS) THEN
        IF(NTERMS.EQ.2) THEN
          D(1)=D(1)-D(2)*B(1)
          RETURN
        ELSE
          GOTO 71
        ENDIF
      ENDIF
      DO 60 I=1,NP
        ERROR(I)=ERROR(I)-D(J)*PJ(I)
 60   CONTINUE
      B(J)=B(J)/S(J)
      C(J)=S(J)/S(J-1)
      DO 70 I=1,NP
        P=PJ(I)
        PJ(I)=(ZZ(I)-B(J))*PJ(I)-C(J)*PJM1(I)
        PJM1(I)=P
 70   CONTINUE
      GOTO 41
 71   D(1)=D(1)-D(2)*B(1)+D(3)*(B(2)*B(1)-C(2))
      D(2)=D(2)-D(3)*(B(2)+B(1))
      RETURN
      END



      SUBROUTINE EXPGRD

C*****************************************************
C     THIS SUBROUTINE IS A TRIDIAGONAL SOLVER FOR THE  *
C     GRID GENERATION EQUATION GIVEN BY LICK AND       *
C     GASKINS                                          *
C*****************************************************

      COMMON /A/IMAX,U(51),ZI(51),X(51),P(51),DUMMY(51),RE
      COMMON /B/KGRID,GRIDACC,XMIN,XMAX
      COMMON /C/A(51),C(51),D(51),G(51),W(51)
      WRITE (6,56)
 56   FORMAT (/T15,'GRID CAL',/,T5,'USING A UNIFIED DIFF REL')
      X(1)=XMIN
      X(IMAX)=XMAX
      DO 190 I=2,IMAX-1
        A(I)=EXP(-P(I))/(EXP(-P(I))+1.0)
        C(I)=1/(EXP(-P(I))+1.0)
        D(I)=0.0
 190  CONTINUE
      G(1)=X(1)
      W(1)=0.0
      DO 290 I=2,IMAX-1
        W(I)=-C(I)/(1.0+A(I)*W(I-1))
        G(I)=(D(I)+A(I)*G(I-1))/(1.+A(I)*W(I-1))
 290  CONTINUE
      DO 390 I=IMAX-1,2,-1
        X(I)=G(I)-W(I)*X(I+1)
```

```
390  CONTINUE
     WRITE (6,80)
80   FORMAT(10X,'TRIDIAGONAL SUBROUTINE IS USED FOR GRID EQ')
     RETURN
     END
```

Bibliography

1. Thompson, J. F., Z. U. A. Warsi, and C. W. Mastin. "Boundary Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations--A Review," Journal of Computational Physics, 47: 1-108 (1982).

2. Thompson, J. F. "A Survey of Grid Generation Techniques in Computational Fluid Dynamics," AIAA Paper No. 83-0447, 1-36 (January 1983).

3. Anderson, D. A. "Adaptive Grid Methods for Partial Differential Equations," Advances in Grid Generation, K. N. Ghia and U. Ghia, eds. ASME FED, 5: 1-15 (1983).

4. Thompson, J. F. "A Survey of Dynamically-Adaptive Grids in the Solution of Partial Differential Equations," AIAA Paper No. 84-1606, 1-15 (June 1984).

5. Hodge, J. K. and A. L. Stone. "Numerical Solution for Airfoils Near Stall in Optimized Boundary Fitted Coordinates," AIAA Paper No. 78-284, 1-12 (January 1978).

6. Thompson, J. F., F. C. Thames, and C. W. Mastin. "Automated Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies," Journal of Computational Physics, 15: 299-319 (July 1974).

7. Dwyer, H. A., R. J. Kee, and B. R. Sanders. "Adaptive Grid Methods for Problems in Fluid Mechanics and Heat Transfer," AIAA Journal, 18: 1205-1212 (October 1980).

8. Pierson, B. L. and P. Kutler. "Optical Nodal Point Distribution for Improved Accuracy in Computational Fluid Dynamics," AIAA Journal, 18: 49-54 (January 1980).

9. Anderson, D. A. and M. M. Rai. "The Use of Solution Adaptive Grids in Solving Partial Differential Equations," Numerical Grid Generation, J. F. Thompson, ed. New York: North Holland, 1982.

10. Brown, K. G. Adaptive Grid Generation for Numerical Solution of Partial Differential Equations. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH (December 1983).

11. Saltzman, J. and J. Brackbill. "Applications and Generalizations of Variational Methods for Generating Adaptive Meshes," Numerical Grid Generation, J. F. Thompson, ed. New York: North Holland, 1982.

12. Ghia, K. N., U. Ghia, and C. T. Shin. "Adaptive Grid Generation for Flows with High Local Gradient Regions," Advances in Grid Generation, K. N. Ghia and U. Ghia, eds. ASME FED, 5: 35-47 (1983).

13. Lick, W. and T. Gaskins. "A Consistent and Accurate Procedure for Obtaining Difference Equations from Differential Equations," International Journal for Numerical Methods in Engineering, 20: 1433-1441 (1984).

14. Conte, S. D. and C. de Boor. Elementary Numerical Analysis: An Algorithmic Approach (Second Edition). New York: McGraw-Hill Book Company, 1972.

Vita

Second Lieutenant Bruce D. Boyd was born 29 January 1961 in Cincinnati, Ohio. He graduated from high school in Cincinnati in 1979. He entered Parks College of St. Louis University in Cahokia, Illinois and accepted an ROTC scholarship. He was awarded a Bachelor of Science degree in Aerospace Engineering in 1982 and received a commission in the United States Air Force. He entered the School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio in May 1983, his first assignment.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A152217

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GAE/AA/84D-2	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/EN	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) Bruce D. Boyd, B.S., 2d Lt, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1984 December 15. PAGE COUNT 89
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Aerodynamics, Fluid Dynamics, Adaptive Systems, Computer Applications.
01	01		
20	04		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: ADAPTIVE GRID GENERATION FOR NUMERICAL SOLUTION OF BURGER'S EQUATION

Thesis Chairman: Sal A. Leone, Captain, USAF

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Sal A. Leone, Captain, USAF	22b. TELEPHONE NUMBER (Include Area Code) 513-255-3708 22c. OFFICE SYMBOL AFIT/ENY

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

An adaptive grid generation method is presented which is based on the reduction of truncation error in the computational plane. Grid adaption is achieved through minimization of the third derivative of the dependent variable in the computational plane. Burger's equation is solved because it represents typical nonlinear fluid-flow equations. An optimized Successive-Over-Relaxation method is used to solve Burger's equation using second-order-upwind differences for the convective term.

Results are presented for several cases which are compared to the results of a test case. The results show grid points are concentrated in high gradient regions.

END

FILMED

5-85

DTIC